Theoretical Computer Science 462 (2012) 39–58

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Chosen-ciphertext secure anonymous conditional proxy re-encryption with keyword search

Liming Fang [a], Willy Susilo [b,*], Chunpeng Ge [a], Jiandong Wang [a]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, 29 Yudao Street, Nanjing, China

[b] Centre for Computer and Information Security Research (CCISR), School of Computer Science and Software Engineering, University of Wollongong, Northfields Avenue, NSW 2522, Australia

ARTICLE INFO

Article history:
Received 7 September 2011
Received in revised form 6 June 2012
Accepted 24 August 2012
Communicated by X. Deng

Keywords:
Public key encryption
Conditional proxy re-encryption
Keyword search
Anonymity
Chosen-ciphertext security

ABSTRACT

Weng et al. introduced the notion of *conditional proxy re-encryption* (or C-PRE, for short), whereby only the ciphertext satisfying one condition set by the delegator can be transformed by the proxy and then decrypted by delegatee. Nonetheless, they left an open problem on how to construct CCA-secure C-PRE schemes with anonymity. Fang et al. answered this question by presenting a construction of anonymous conditional proxy re-encryption (C-PRE) scheme without requiring random oracle. Nevertheless, Fang et al.'s scheme only satisfies the RCCA-security (which is a weaker variant of CCA-security assuming a harmless mauling of the challenge ciphertext is tolerated). Hence, it remains an open problem whether CCA-secure C-PRE schemes that satisfy *both* anonymity and full CCA-security can really be realized. Shao et al. introduced a new cryptographic primitive, called proxy re-encryption with keyword search (PRES), which is a combination of PRE and public key encryption with keyword search (PEKS), and they left an open problem on how to design an efficient unidirectional PRES scheme.

In this paper, we answer the above open problems by proposing a new cryptographic primitive called conditional proxy re-encryption with keyword search (C-PRES), which combines C-PRE and PEKS. We note that there are subtleties in combining these two notions to achieve a secure scheme, and hence, the combination is not trivial. We propose a definition of security against chosen ciphertext attacks for C-PRES schemes with keyword anonymity, and thereafter present a scheme that satisfies the definition. The performance of our scheme outperforms Weng et al.'s construction, which has been regarded as the most efficient C-PRE scheme to date.

1. Introduction

Nowadays, more and more users store their private data in cloud. To ensure the security of the remotely stored data, the user needs to encrypt the private data under her public key. However, users usually do not retrieve all the encrypted data but part of them, which demands a searchable encryption scheme supporting a keyword-based search on the ciphertext. Consider the following scenario. Suppose all of the data of a company have been stored in the cloud. Alice is the HR manager of this company. The content of Alice's data comprises a body of the data and a keyword that are encrypted under Alice's public key. In this case, the cloud provider cannot observe the information of the data including the keyword and message since we need to ensure the privacy of the shared data. Bob is the director of this company. He is only interested in Alice's data

* Corresponding author. Tel.: +61 2 4221 5535; fax: +61 2 4221 4170.
E-mail addresses: fangliming@nuaa.edu.cn (L. Fang), wsusilo@uow.edu.au (W. Susilo), gecp@nuaa.edu.cn (C. Ge), aics@nuaa.edu.cn (J. Wang).

0304-3975/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.08.017

with the keyword $w =$ "interview result" and furthermore, Alice also only wants Bob to read her data (*i.e.*, the ciphertexts) satisfying the keyword $w =$ "interview result" rather than all of her ciphertexts. Additionally, both Alice and Bob do not want to let the cloud provider know that actually they are somewhat *sharing* the data with the keyword $w =$ "interview result". In other words, the *privacy* of the keyword itself is important, as exposing this keyword may result in leaking the important information to the competitors (for example, the cloud can be used by many other companies, including Alice and Bob's competitors).

To address this issue, Shao et al. [31] proposed a new cryptographic primitive, called proxy re-encryption with keyword search (PRES), by combining the notions of PRE and PEKS directly, which means

$$PRE(pk_A, m)||PEKS(pk_A, w)$$

where pk_A is Alice's public key, and w is the keyword with the message m. On the one hand the cloud provider with a trapdoor can search the desirable ciphertexts, while the cloud provider cannot obtain the keyword. On the other hand, the cloud provider with a re-encryption key can re-encrypt the ciphertext under Alice's public key to create another ciphertext under Bob's public key, while the cloud provider cannot learn the plaintext and the keyword.

We note that although Shao et al.'s [31] solution for realizing PRES is elegant, there remain some important issues regarding the use of PRES, which have not been addressed in their paper. The issues are the following:

- First, Shao et al.'s work addresses only bidirectional cases, while it is more desirable to find a solution for a unidirectional case. They also leave an open problem on how to design an efficient unidirectional PRES scheme.
- Additionally, a proxy in Shao et al.'s scheme is too powerful as it has the ability to encrypt *all* Alice's emails to Bob once the re-encryption key is provided.
- Furthermore, their scheme is bidirectional and hence, it inherently fails to provide collusion-resistance. Consequently, the proxy with the bidirectional re-encryption key $rk_{i,j} = x_j/x_i$ and the delegatee with the private key $sk_j = x_j$ can collude together to expose the delegator's private key $sk_i = x_i$. This problem cannot be solved by a simple modification.

We note that Yau et al. [37] also proposed a proxy re-encryption with keyword search scheme (Re-(d)PEKS) with a designated tester, but their scheme is *not* collusion-safe and unidirectional, and it is difficult to integrate both Re-(d)PEKS and PRE.

In this paper, instead of extending the notion of PRE, we incorporate the notion of conditional proxy re-encryption (C-PRE) whereby the ciphertexts satisfying a condition (*i.e.,* a certain keyword) by Alice can be transformed by the cloud provider and then, decrypted by Bob. In other words, we propose a new cryptographic primitive, called conditional proxy re-encryption with keyword search (C-PRES), by combining C-PRE and PEKS, which means

$$C - PRE(pk_A, m, w)||PEKS(pk_A, w)$$

where pk_A is Alice's public key, and w is the keyword with the message m. As discussed in [4,39,40], it is noted that a trivial combination of these two notions will result in an insecure scheme. For instance, a naive composition of a stand-alone secure PEKS and a CCA secure C-PRE may lose data privacy due to the chosen ciphertext attack: when an adversary observes a C-PRE/PEKS ciphertext $CT = C - PRE(pk_A, m, w)||PEKS(pk_A, w)$, it can produce another valid ciphertext $CT' = C - PRE(pk_A, m, w)||PEKS(pk_A, w')$. Querying CT' to a decryption oracle, the adversary obtains the plaintext m. Besides, C-PRE may leak the information of keyword w since it does not satisfy keyword-anonymity. Therefore, cautions must be exercised when combining these two notions. Hence, we need to address the subtleties in combining the two.

By studying the definitions and security notions for previous C-PRE and PRES, there remain some important issues to consider:

- (*Keyword-anonymity.*) In the previous C-PRE schemes, the ciphertext will leak the information of the keyword, and therefore Weng et al. [33] left an interesting open problem on how to construct CCA-secure C-PRE schemes *with anonymity*. This is also essential for C-PRES scheme to keep the keyword anonymity.
- (*Chosen-Ciphertext Security.*) Fang et al.'s [16] anonymous C-PRE scheme only satisfies the RCCA-security which is a weaker variant of CCA-security assuming a harmless mauling of the challenge ciphertext is tolerated. It is even difficult to construct CCA-secure C-PRE scheme with anonymity, let alone to create the CCA-secure C-PRES scheme.
- (*First and second level ciphertext security.*) Both of the security notions in [33] and [32] only considered the second level ciphertext security, and do not address the first level ciphertext security, while the work in [23,34] take into account both the first and second level ciphertext security.
- (*Unidirectionality.*) In a bidirectional PRE, the proxy can transform from a delegator to a delegatee and vice versa. In contrast, the proxy in the unidirectional PRE cannot transform ciphertexts in the opposite direction. The literature has demonstrated that a bidirectional scheme is much easier to design. Thus, Shao et al. [31] left an open problem on how to design an efficient unidirectional PRES scheme.
- (*Non-interactivity.*) The delegatee does not act in the delegation process.
- (*Collusion-resistance.*) It is impossible to export the delegator's private key when the dishonest proxy colludes with the delegatee.

1.1. Our contributions

As discussed above, it is non-trivial to construct a C-PRES scheme since a C-PRES scheme requires an anonymous IBE, and the ciphertext of the anonymous IBE can be re-encrypted. It is much more difficult to achieve chosen-ciphertext security while not jeopardizing the properties of keyword-anonymity, unidirectionality, non-interactivity and collusion-resistance.

In this paper, we aim to address the open problems on how to construct CCA-secure C-PRE schemes *with anonymity* and how to design an efficient unidirectional PRES scheme. Concretely, we formalize the security model of conditional proxy re-encryption with keyword search (C-PRES) scheme. Then, we present an efficient construction of C-PRES scheme that offers several advantages over previous such systems, including: chosen-ciphertext security; keyword-anonymity; unidirectionality; non-interactivity; and collusion-resistance.

Our scheme outperforms Shao et al.'s PRES scheme in terms of both computational and communicational costs. Furthermore, our scheme is collusion-resistant and it is a conditional re-encryption scheme, while Shao et al.'s PRES scheme is bidirectional only. Compared with Fang et al.'s anonymous C-PRE [16], our scheme is also superior [16] in the following aspects: (i) In contrast to Fang et al.'s scheme, our scheme provides CCA-security; (ii) Our scheme has better overall efficiency in terms of both computation and communication cost.

1.2. Related work

Proxy re-encryption

The concept of proxy re-encryption (PRE) dates back to the work of Blaze et al. [5] in 1998. The goal of such systems is to securely enable the re-encryption of ciphertexts from one key to another, without relying on trusted parties. PRE can be categorized into bidirectional PRE and unidirectional PRE. In a bidirectional PRE, the proxy can transform from delegator to delegatee and vice versa. In contrast, the proxy in a unidirectional PRE cannot transform ciphertexts in the opposite direction. In 2005, Ateniese et al. [1] demonstrated how to construct unidirectional schemes using bilinear maps and simultaneously prevent proxies from colluding with delegatees in order to expose the delegator's secret key. In 2006, Green and Ateniese [19] extended the above notion to identity-based proxy re-encryption (IB-PRE), and proposed new CCA secure scheme. In 2007, Canetti and Hohenberger [9] also proposed a new CCA secure PRE scheme where the proxy can verify the validity of the ciphertext prior to the transformation. In 2007, Chu and Tzeng [12] proposed a multi-use, unidirectional ID-based PRE scheme, and claimed that it was CCA secure in the standard model. In PKC 08, Libert and Vergnaud [23] presented a replayable chosen-ciphertext (RCCA) secure single-hop unidirectional proxy re-encryption scheme in the standard model. Hohenberger et al. [21] developed an obfuscated re-encryption program which translates ciphertexts under pk_A to ciphertexts under pk_B. Due to the fact that pairing computation is a costly operation, the subsequent work [15,30,14, 26] focused on PRE schemes constructed without bilinear pairings, especially in computation resource limited settings.

Conditional proxy re-encryption

Instead of converting all ciphertexts, Alice may only want the proxy to convert the ciphertexts with a certain keyword, such as "business". To address this problem, Weng et al. [33] presented a notion of conditional proxy re-encryption (C-PRE), whereby only ciphertexts satisfying a condition set by Alice can be transformed by the proxy and then decrypted by Bob. They also proposed an efficient CCA secure C-PRE scheme in the random oracle model. Unfortunately, Weng et al. [34] demonstrated that Weng et al.'s C-PRE scheme [33] failed to achieve the CCA-security, and they further formalized a more rigorous definition and proposed a more efficient CCA secure C-PRE scheme in the random oracle model. Tang et al. [32] introduced type-based proxy re-encryption. Actually, the construction of conditional PRE scheme had already been proposed [24] in 2008. In their work [24], Libert and Vergnaud suggested a PRE scheme which provided warrant-based and keyword-based delegations without random oracle. Recently, Chu et al. introduced a conditional proxy broadcast re-encryption [13], in which the proxy can re-broadcast ciphertexts for a set of users.

Furthermore, Weng et al. [33] left an interesting open problem on how to construct CCA-secure C-PRE schemes *with anonymity*. To fill this gap, Fang et al. first formalized the notion of *anonymous conditional re-playable chosen-ciphertext attacks (RCCA) secure PRE* and presented a respective security model [16]. Then, they presented a construction of anonymous C-PRE scheme without requiring random oracle. Canetti et al. [11] introduced an approach on how to generically turn any RCCA secure PKE scheme into a CCA secure PKE scheme. Their idea is as follows.

Given a RCCA secure public-key encryption scheme (Gen, Enc, Dec) and CCA secure symmetric encryption scheme (E, D), let the function $(e, d) = Gen(r)$, where r is the random bits used in generating (e, d), and let l be the security parameter. Then the new CCA secure PKE scheme $(\overline{Gen}, \overline{Enc}, \overline{Dec})$ is

$$\overline{Enc}_e(m) = [K \leftarrow \{0,1\}^l; c_1 = Enc_e(K); c_2 = E_K(c_1 \| m) : (c_1, c_2)],$$
$$\overline{Dec}_d(c_1, c_2) = [K \leftarrow Dec_d(c_1); c_1' \| m = D_K(c_2); \textit{if } c_1' \neq c_1 \textit{ then } m \leftarrow \textit{invalid} : m].$$

This encryption of c_1 functions as a *MAC* which protects against "mauling" of c_1. The user can check the validity of the ciphertext only when he can decrypt the symmetric key K. But the transformation does not make sense while to transform RCCA secure PRE scheme to CCA one, after transforming it, the second level encryption would be $CT_i = (c_1, c_2) = (c_1 = Enc2_{sk_i}(K), c_2 = E_K(c_1 \| m))$ and the first level ciphertext $CT_j = (c_3, c_2)$ where $c_3 = ReEnc(c_1)$. When an adversary observes a new second level ciphertext $CT_i = (c_1, c_2)$, it can produce another valid second level ciphertext $CT_i' = (c_1, c_2')$ where c_2'

is randomly chosen. Clearly, the proxy cannot check the validity of the ciphertext $CT_i' = (c_1, c_2')$ since he cannot decrypt the symmetric key K. Querying CT_i' to a re-encryption oracle to get the first level ciphertext $CT_j' = (c_3, c_2')$, the adversary obtains the plaintext m after querying $CT_j'' = (c_3, c_2)$ to a decryption oracle.

Public key encryption with keyword search

Following Boneh et al.'s pioneering work [7], Waters et al. [35] demonstrated that the PEKS scheme based on the bilinear pairing can be applied to build encrypted and searchable audit logs. Baek et al. [3] and Rhee et al. [27] improved the public key encryption with keyword search scheme in [7]. Furthermore, Golle et al. [20] and Park et al. [25] proposed schemes that allow conjunctive keyword queries on encrypted data. Recently, Zhang and Zhang [38] proposed a more efficient construction of public key encryption with conjunctive-subset keywords search scheme. Moreover, the subsequent papers [4,39] investigated the secure combination of public key encryption with keyword search (PEKS) with public key data encryption (PKE). Because of the fact that keywords are chosen from much smaller space than passwords and users usually use well-known keywords for search, the research reported in [8,22,28,29,36] studied the off-line keyword guessing attacks on PEKS.

1.3. Paper organization

The rest of this paper is organized as follows. In Section 2, we will provide the definitions and complexity assumption that will be used throughout this paper, together with the security model of anonymous C-PRES schemes. In Section 3, we present our anonymous C-PRES in the random oracle model. In Section 4, we compare our scheme with previously reported C-PRE and PRES schemes. Finally, Section 5 concludes the paper.

2. Definitions

In this section, we first review the complexity assumption required in our schemes, and then provide the definition and security of a conditional proxy re-encryption with keyword search (C-PRES) scheme.

2.1. Negligible function

A function $\epsilon(n) : N \mapsto R$ is negligible in n if $1/\epsilon(n)$ is a non-polynomially-bounded quantity in n.

2.2. Taylor's formula

For completeness, we review Taylor's formula in this section. We shall incorporate this formula in our concrete construction.

Let $k \geq 1$ be an integer and let the function $f : R \mapsto R$ be k times differentiable at the point $a \in R$. Then there exists a function such that $f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^k + R_k(x)$.

For the Lagrange form of the remainder [2], there exists ξ_L between a and x. Therefore, $R_k(x) = \frac{f^{(k+1)}(\xi_L)}{(k+1)!}(x-a)^{(k+1)}$. Thus, we have

$$\begin{aligned}\frac{f(x)-f(a)}{x-a} &= f'(a) + \frac{f''(a)}{2!}(x-a) + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^{k-1} + \frac{f^{(k+1)}(\xi_L)}{(k+1)!}(x-a)^k \\ &= a_0 + a_1 x + \cdots + a_{k-1}x^{k-1} + a_k x^k.\end{aligned}$$

2.3. Bilinear maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be multiplicative cyclic groups of prime order p, and g be a generator of $\mathbb{G}_1$. We say $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map [6], if the following conditions hold.

(1) $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $a, b \in \mathbb{Z}_p$ and $g_1, g_2 \in \mathbb{G}_1$.
(2) $e(g, g) \neq 1$.
(3) There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}_1$.

2.4. The DBDH assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. We define the advantage function $Adv_{\mathbb{G}_1,\mathcal{B}}^{DBDH}(\lambda)$ of an adversary $\mathcal{B}$ as

$$|Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] - Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^r) = 1]|$$

where $a, b, c, r \in \mathbb{Z}_p$ are randomly chosen. We say that the decisional bilinear Diffie Hellman assumption [6] relative to generator $\mathbb{G}_1$ holds if $Adv_{\mathbb{G}_1,\mathcal{B}}^{DBDH}(\lambda)$ is negligible for all PPT $\mathcal{B}$.

2.5. The truncated q-ABDHE assumption

Let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a bilinear map. We define the advantage function $Adv^{q-ABDHE}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ of an adversary $\mathcal{B}$ as

$$|Pr[\mathcal{B}(g, g^x, \ldots, g^{x^q}, g^z, g^{zx^{q+2}}, e(g,g)^{zx^{q+1}}) = 1] - Pr[\mathcal{B}\ \ (g, g^x, \ldots, g^{x^q}, g^z, g^{zx^{q+2}}, e(g,g)^r) = 1]|$$

where $x, z, r \in \mathbb{Z}_p$ are randomly chosen. We say that the truncated q-ABDHE assumption [18] relative to generator $\mathbb{G}_1$ holds if $Adv^{q-ABDHE}_{\mathbb{G}_1,\mathcal{B}}(\lambda)$ is negligible for all PPT $\mathcal{B}$.

2.6. One-time signatures

A one-time signature [10] comprises a triple of algorithms $sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ such that, on input of a security parameter λ, $\mathcal{G}$ generates a one-time key pair (ssk, svk) where $k_1 = |svk|$ while, for any message M, $\mathcal{V}(svk, \sigma, M)$ outputs 1 whenever $\sigma = \mathcal{S}(ssk, M)$ and 0, otherwise. We need strongly unforgeable one-time signatures, which means that no PPT adversary can create a new signature for a previously signed message.

$sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature if the probability

$$Adv^{OTS} = Pr[(ssk, svk) \leftarrow \mathcal{G}(\lambda); (M, St) \leftarrow F(svk); \sigma \leftarrow \mathcal{S}(ssk, M);$$
$$(M', \sigma') \leftarrow F(M, \sigma, svk, St) : \mathcal{V}(svk, \sigma', M') = 1 \wedge (M', \sigma') \neq (M, \sigma)]$$

where St denotes the state information maintained by F between stages, is negligible for any PPT forger F.

2.7. Conditional proxy re-encryption with keyword search

In the following, we will provide the definition of a conditional proxy re-encryption with keyword search scheme and the game-based security definition.

Definition 1 (*Conditional Proxy Re-encryption with Keyword Search*)**.** A (single hop) conditional proxy re-encryption with keyword search scheme comprises the following algorithms:

- *GlobalSetup*(λ): The GlobalSetup algorithm is run by a trusted party that takes as input a security parameter λ. It generates the global parameters *PP*.
- *KeyGen*(i): The key generation algorithm generates the public key pk_i and the secret key sk_i for user i.
- *RKeyGen*(pk_i, sk_i, w, pk_j): The re-encryption key generation algorithm, run by user i, takes as input a public key pk_i, a secret key sk_i, a condition w and another public key pk_j. It outputs a re-encryption key $rk_{i,w,j}$.
- *Trapdoor*(pk_i, sk_i, w): The trapdoor generation algorithm, run by user i, takes as input a public key pk_i, a secret key sk_i and a condition w. It outputs a trapdoor $T_{i,w}$.
- *Enc*1(pk, m): The level 1 encryption algorithm takes as input a public key pk, and a plaintext $m \in \mathcal{M}$. It outputs a first level ciphertext CT under public key pk. Here $\mathcal{M}$ denotes the message space.
- *Enc*2(pk, m, w): The level 2 encryption algorithm takes as input a public key pk, a plaintext $m \in \mathcal{M}$ and a condition w. It outputs a second level ciphertext CT associated with w under public key pk.
- *Test*(CT_i, $T_{i,w}$): The Test algorithm, run by the proxy, takes as input a second level ciphertext CT_i associated with w' under public key pk_i, and a trapdoor $T_{i,w}$. It outputs "1" if $w = w'$ and "0" otherwise.
- *ReEnc*(CT_i, $rk_{i,w,j}$): The re-encryption algorithm, run by the proxy, takes as input a second level ciphertext CT_i associated with w under public key pk_i, and a re-encryption key $rk_{i,w,j}$. It outputs the first ciphertext (level 1) CT_j under the public key pk_j, or an error symbol $\perp$.
- *Dec*1(CT_j, sk_j): The level 1 decryption algorithm takes as input a secret key sk_j and a first level ciphertext CT_j under public key pk_j. It outputs a message $m \in \mathcal{M}$ or an error symbol $\perp$.
- *Dec*2(CT_i, sk_i): The level 2 decryption algorithm takes as input a secret key sk_i and a second level ciphertext CT_i. It outputs a message $m \in \mathcal{M}$ or an error symbol $\perp$.

Note that we omit the global parameters *PP* as the other algorithms' input for simplicity. The correctness of C-PRES means that, a correctly generated ciphertext can be correctly decrypted by the user who has the correct secret key, *i.e.*, for any condition w, any message m, any $(pk_i, sk_i) \leftarrow KeyGen(i)$, $(pk_j, sk_j) \leftarrow KeyGen(j)$, and $CT_i = Enc2(pk, m, w)$,

$$Pr[Dec2(CT_i, sk_i) = m] = 1$$

and

$$Pr[Dec1(ReEnc(CT_i, RKeyGen(pk_i, sk_i, w, pk_j)), sk_j) = m] = 1.$$

In the following, we provide the game-based security definition of C-PRES. As in [31], we consider the privacy for message and privacy for keyword. For the former, the adversary is allowed to get the plaintexts of almost all ciphertexts except for a specified ciphertext. The latter security notion guarantees that the adversary can acquire any trapdoors, except the ones that

are associated with the specified keywords, and further, it should not be able to decide which keyword corresponds to the provided ciphertext. This security notion guarantees that only the one who has the private key can decrypt ciphertexts. We divide it into two level security: security of second level ciphertexts and security of first level ciphertexts. For the latter, the adversary is allowed to get the plaintext of any ciphertext, and almost all trapdoors except those which are associated with the two specified keywords, however, it cannot decide which keyword corresponds to the given ciphertext. This security notion guarantees that only the one who has the trapdoor can do the test.

Additionally, our definition considers a challenger that produces a number of public keys. As in [23], we let the corrupted users and the honest users be determined at the beginning of the game. Furthermore, we allow the adversary to adaptively query a re-encryption oracle and decryption oracles.

Definition 2 (*C-PRES-IND-ANON-CCA game*)**.** Let λ be the security parameter and $\mathcal{A}$ be the adversary. We consider the following two games.

Game 1: (IND-ANON game: Privacy for keyword.)

(1) Setup: The challenger $\mathcal{C}$ performs $GlobalSetup(\lambda)$ to get the public parameter PP. Give the public parameter PP to $\mathcal{A}$.
(2) Query phase 1. $\mathcal{A}$ makes the following queries:
- Uncorrupted key generation query $\langle i\rangle$: $\mathcal{C}$ first runs algorithm $KeyGen(i)$ to obtain a public/secret key pair (pk_i, sk_i), and then sends pk_i to $\mathcal{A}$.
- Corrupted key generation query $\langle j\rangle$: $\mathcal{C}$ first runs algorithm $KeyGen(j)$ to obtain a public/secret key pair (pk_j, sk_j), and then sends (pk_j, sk_j) to $\mathcal{A}$.
- Re-encryption key query $\langle pk_i, w, pk_j\rangle$: $\mathcal{C}$ runs algorithm $RKeyGen(pk_i, sk_i, w, pk_j)$ to generate a re-encryption key $rk_{i,w,j}$ and returns it to $\mathcal{A}$. Here, sk_i is the secret key with respect to pk_i. Here, different from [31], we allow the re-encryption key generation queries between a corrupted key and an uncorrupted key.
- Trapdoor query $\langle pk_i, w\rangle$: $\mathcal{C}$ runs algorithm $Trapdoor(pk_i, sk_i, w)$ to generate a trapdoor $T_{i,w}$ and returns it to $\mathcal{A}$.
- Test query $\langle pk_i, w, CT_i\rangle$: $\mathcal{C}$ runs algorithm $Test(CT_i, Trapdoor(pk_i, sk_i, w))$ where sk_i is the secret key corresponding to pk_i and returns the result to $\mathcal{A}$.
- Re-encryption query$\langle pk_i, pk_j, (w, CT_i)\rangle$: $\mathcal{C}$ runs algorithm

$$CT_j = ReEnc(CT_i, RKeyGen(pk_i, sk_i, w, pk_j))$$

and returns the resulting ciphertext CT_j to $\mathcal{A}$. It is required that pk_i and pk_j have been generated beforehand by algorithm $KeyGen$.
- Decryption query $\langle pk_i, (w, CT_i)\rangle$: Here $\langle pk_i, (w, CT_i)\rangle$ denotes the queries on second level ciphertext (level 2). Challenger $\mathcal{C}$ returns the result of $Dec2(CT_i, sk_i)$ to $\mathcal{A}$. It is required that pk_i has been generated beforehand by algorithm $KeyGen$.
- Decryption query $\langle pk_j, CT_j\rangle$: Here $\langle pk_j, CT_j\rangle$ denotes the queries on re-encrypted ciphertext (level 1). Challenger $\mathcal{C}$ returns the result of $Dec1(CT_j, sk_j)$ to $\mathcal{A}$. It is required that pk_j has been generated beforehand by algorithm $KeyGen$.

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, a condition pair (w_0, w_1) and a plaintext m. Challenger $\mathcal{C}$ chooses a bit $\beta \in \{0, 1\}$ and sets the challenge ciphertext to be $CT^* = Enc2(pk_{i^*}, m, w_\beta)$, which is sent to $\mathcal{A}$.
(4) Query phase 2. $\mathcal{A}$ continues making queries as in the query phase 1.
(5) Guess. $\mathcal{A}$ outputs the guess β'. The adversary wins if $\beta' = \beta$.

During the above game, adversary $\mathcal{A}$ is subject to the following restrictions where $w^* \in \{w_0, w_1\}$:

- (i). $\mathcal{A}$ cannot issue corrupted key generation queries on $\langle i^*\rangle$ to obtain the target secret key sk_{i^*}.
- (ii). $\mathcal{A}$ cannot obtain the trapdoor query on $\langle pk_{i^*}, w^*\rangle$. Otherwise, the adversary can win the IND-ANON game trivially.
- (iii). $\mathcal{A}$ cannot obtain the test query on $\langle pk_{i^*}, w^*, CT^*\rangle$.
- (iv). $\mathcal{A}$ cannot issue decryption queries on neither $\langle pk_{i^*}, (w^*, CT^*)\rangle$ nor $\langle pk_j, CT_j^*\rangle$ where $\langle pk_j, CT_j^*\rangle$ is a re-encryption of the challenge pair $\langle pk_{i^*}, (w^*, CT^*)\rangle$.
- (v). $\mathcal{A}$ cannot issue re-encryption queries on $\langle pk_{i^*}, pk_j, (w^*, CT^*)\rangle$ if pk_j appears in a previous corrupted key generation query.
- (vi). $\mathcal{A}$ cannot obtain the re-encryption key $rk_{i^*,w^*,j}$, if pk_j appears in a previous corrupted key generation query.

We refer to the above adversary $\mathcal{A}$ as an IND-ANON adversary. His advantage is defined as

$$Succ_{\mathcal{A}}^{Game_1}(\lambda) = |Pr[\beta' = \beta] - 1/2|.$$

Game 2: (IND-L2-CCA game: security of level 2 ciphertexts.)

(1) Setup: The challenger $\mathcal{C}$ performs $GlobalSetup(\lambda)$ to get the public parameter PP. Give the public parameter PP to $\mathcal{A}$.
(2) Query phase 1. Identical to that in the security model of Game 1.
(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, a condition keyword w^* and two equal length plaintexts (m_0, m_1). Challenger $\mathcal{C}$ chooses a bit $\beta \in \{0, 1\}$ and sets the challenge ciphertext to be $CT^* = Enc2(pk_{i^*}, m_\beta, w^*)$, which is sent to $\mathcal{A}$.

(4) Query phase 2. $\mathcal{A}$ continues making queries as in the query phase 1.
(5) Guess. $\mathcal{A}$ outputs the guess β'. The adversary wins if $\beta' = \beta$.

During the above game, adversary $\mathcal{A}$ is subject to the following restrictions:

- (i). $\mathcal{A}$ cannot issue corrupted key generation queries on $\langle i^* \rangle$ to obtain the target secret key sk_{i^*}.
- (ii). $\mathcal{A}$ cannot issue decryption queries on neither $\langle pk_{i^*}, (w^*, CT^*) \rangle$ nor $\langle pk_j, CT_j^* \rangle$ where $\langle pk_j, CT_j^* \rangle$ is a re-encryption of the challenge pair $\langle pk_{i^*}, (w^*, CT^*) \rangle$.
- (iii). $\mathcal{A}$ cannot issue re-encryption queries on $\langle pk_{i^*}, pk_j, (w^*, CT^*) \rangle$ if pk_j appears in a previous corrupted key generation query.
- (iv). $\mathcal{A}$ cannot obtain the re-encryption key $rk_{i^*,w^*,j}$, if pk_j appears in a previous corrupted key generation query.

We refer to the above adversary $\mathcal{A}$ as an IND-L2-CCA adversary. His advantage is defined as

$$Succ_{\mathcal{A}}^{Game_2}(\lambda) = |Pr[\beta' = \beta] - 1/2|.$$

Game 3: (IND-L1-CCA: Security of level 1 ciphertexts.) Next, we will consider the definition of security of level 1 by providing the adversary with a level 1 ciphertext in the challenge phase. For single-hop schemes, the adversary is provided with access to all re-encryption keys and trapdoors in this definition. The re-encryption and test oracles thus become useless since $\mathcal{A}$ can re-encrypt ciphertexts or test by himself when given all re-encryption keys and trapdoors. Thus, a level 2 decryption is also unnecessary. As in [14], not only the security of a non-transformable ciphertext with a challenge (which is defined as $CT^* = Enc1(pk_{i^*}, m_\beta)$, i.e., Non-transformable Ciphertext Security), that was considered, but we also transformed the ciphertext with a challenge defined as $CT^* = ReEnc(Enc2(pk_{i'}, m_\beta, w), rk_{i',w,i^*})$ (i.e., Transformed Ciphertext Security). Thus, the challenge ciphertext would be the form of $CT^* = Enc1(pk_{i^*}, m_\beta)$ or $CT^* = ReEnc(Enc2(pk_{i'}, m_\beta, w), rk_{i',w,i^*})$.

(1) Setup: The challenger $\mathcal{C}$ performs $GlobalSetup(\lambda)$ to get the public parameter PP. Give the public parameter PP to $\mathcal{A}$.
(2) Query phase 1. $\mathcal{A}$ makes the following queries:
   - Uncorrupted key generation query $\langle i \rangle$: $\mathcal{C}$ first runs algorithm $KeyGen(i)$ to obtain a public/secret key pair (pk_i, sk_i), and then sends pk_i to $\mathcal{A}$.
   - Corrupted key generation query $\langle j \rangle$: $\mathcal{C}$ first runs algorithm $KeyGen(j)$ to obtain a public/secret key pair (pk_j, sk_j), and then sends (pk_j, sk_j) to $\mathcal{A}$.
   - Re-encryption key query$\langle pk_i, w, pk_j \rangle$: $\mathcal{C}$ runs algorithm $RKeyGen(pk_i, sk_i, w, pk_j)$ to generate a re-encryption key $rk_{i,w,j}$ and returns it to $\mathcal{A}$. Here, sk_i is the secret key with respect to pk_i.
   - Trapdoor query$\langle pk_i, w \rangle$: $\mathcal{C}$ runs algorithm $Trapdoor(pk_i, sk_i, w)$ to generate a trapdoor $T_{i,w}$ and returns it to $\mathcal{A}$.
   - Decryption query $\langle pk_j, CT_j \rangle$: Here $\langle pk_j, CT_j \rangle$ denotes the queries on re-encrypted ciphertext (level 1). Challenger $\mathcal{C}$ returns the result of $Dec1(CT_j, sk_j)$ to $\mathcal{A}$. It is required that pk_j has been generated beforehand by algorithm $KeyGen$.
(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*} and two equal length plaintexts (m_0, m_1). Challenger $\mathcal{C}$ chooses a bit $\beta \in \{0, 1\}$ and sets the challenge ciphertext to be $CT^* = Enc1(pk_{i^*}, m_\beta)$ (i.e., Non-transformable Ciphertext Security) or $CT^* = ReEnc(Enc2(pk_{i'}, m_\beta, w), rk_{i',w,i^*})$ (i.e., Transformed Ciphertext Security), which is sent to $\mathcal{A}$.
(4) Query phase 2. $\mathcal{A}$ continues making queries as in the query phase 1.
(5) Guess. $\mathcal{A}$ outputs the guess β'. The adversary wins if $\beta' = \beta$.

During the above game, adversary $\mathcal{A}$ is subject to the following restrictions:

- (i). $\mathcal{A}$ cannot issue corrupted key generation queries on $\langle i^* \rangle$ to obtain the target secret key sk_{i^*}.
- (ii). $\mathcal{A}$ cannot issue decryption queries on $\langle pk_{i^*}, CT^* \rangle$.

We refer to the above adversary $\mathcal{A}$ as an IND-L1-CCA adversary. His advantage is defined as

$$Succ_{\mathcal{A}}^{Game_3}(\lambda) = |Pr[\beta' = \beta] - 1/2|.$$

The C-PRES scheme is said to be *C-PRES-IND-ANON-CCA secure* if $Succ_{\mathcal{A}}^{Game_1}(\lambda)$, $Succ_{\mathcal{A}}^{Game_2}(\lambda)$ and $Succ_{\mathcal{A}}^{Game_3}(\lambda)$ are all negligible.

3. Proposed CCA-secure anonymous C-PRES scheme

In this section, inspired by Gentry's IBE scheme [18], we will present our construction of anonymous conditional proxy re-encryption with keyword search scheme with CCA security. Recall that we wish to create a C-PRES scheme in which a ciphertext created under condition w can be tested by the trapdoor and then can be re-encrypted only by a re-encryption key for condition w. In addition, our C-PRES must provide several advantages over previous such systems, including: chosen-ciphertext security; keyword-anonymity; unidirectionality; non-interactivity; and collusion-resistance.

3.1. Our construction

Prior to presenting our scheme, we first present the intuition behind our construction. We select Gentry's IBE scheme in [18] as the initial scheme to work with due to the following reason. After using the keyword to replace the identity in Gentry's IBE scheme in [18], we obtain the second level (original) ciphertext $((X_i g^{-w})^r, e(g,g)^r, e(g, Y_{i,1})^r \cdot m)$, and the user using the private key $((Y_{i,1} g^{-s_1})^{1/(x_i - w)}, s_1)$ under keyword w can decrypt the second level ciphertext. To re-encrypt the second level ciphertext, we change the private key to re-encryption key $((Y_{j,1} Y_{i,1}^{-1} g^{-s_1})^{1/(x_i - w)}, s_1)$. Then, using this key as the re-encryption key will result in the encrypted data under user j, say $((e(g,g)^r, e(g, Y_{j,1})^r \cdot m)$. There are two reasons why we select "Exponent Inversion" IBE, such as the Gentry's IBE scheme. The first reason is Gentry's IBE has the advantage of the identity-anonymity property. The second reason is for "Exponent Inversion" IBE, the principle is to obtain a session key of the form $e(g, Y_{i,1})^r$ based on a ciphertext $g^{f(ID)r}$ and a private key $(Y_{i,1} g^{-s_1})^{1/f(ID)}$, where $f(ID) = x_i - ID$ is a secret function of the recipient identity but $g^{f(ID)}$ is computable publicly. Actually, the re-encryption key $((Y_{j,1} Y_{i,1}^{-1} g^{-s_1})^{1/(x_i - w)}, s_1)$ is protected by $1/f(w)$ where $f(w) = x_i - w$, thus the scheme is collusion-safe and unidirectional, and it is further non-interactive because it cannot involve the private key of user j.

Chosen-ciphertext security. The biggest challenge would be how to achieve the chosen-ciphertext security while not jeopardizing the properties of keyword-anonymity, unidirectionality, non-interactivity and collusion-resistance. To obtain the first level ciphertext chosen-ciphertext security, we use the Fujisaki–Okamoto transformation [17] since the part of $e(g, Y_{i,1})^r \cdot R$ (which needs to be re-encrypted) cannot be fixed, then the form of second level (original) ciphertext is $((X_i g^{-w})^r, e(g,g)^r, e(g, Y_{i,1})^r \cdot R, m \oplus H_2(R))$ and the form of first level (re-encrypted) ciphertext is $(e(g,g)^r, e(g, Y_{i,1})^r \cdot R, m \oplus H_2(R))$ where $r = H_1(m, R)$. To search for the keyword, we can add the term $e(g, Y_{i,3})^r$ to the second level ciphertext, thus the proxy with the trapdoor $((Y_{i,3} g^{-s_3})^{1/(x_i - w)}, s_3)$ can search the keyword. Therefore, the form of second level (original) ciphertext is

$$(C_1, C_2, C_3, C_4, C_5) = ((X_i g^{-w})^r, e(g,g)^r, e(g, Y_{i,1})^r \cdot R, m \oplus H_2(R), e(g, Y_{i,3})^r).$$

To achieve the chosen-ciphertext security of the second level ciphertext, such as [34], one can add the term $H(C_1, C_2, C_3, C_4, C_5)^r$ to ensure the public verifiability of the second level ciphertext. Unfortunately it does not work since this will jeopardize the properties of keyword-anonymity. Another approach is identical to Gentry's CCA secure IBE. It uses a pair of keys to perform ciphertext validity test, then the form of second level (original) ciphertext is

$$(C_1, C_2, C_3, C_4, C_5) = ((X_i g^{-w})^r, e(g,g)^r, e(g, Y_{i,1})^r \cdot R, m \oplus H_2(R), e(g, Y_{i,3})^{r\phi} e(g, Y_{i,4})^r),$$

where $\phi = H_3(C_1, C_2, C_3, C_4)$. Clearly, the trapdoor would be

$$T_{i,w} = ((Y_{i,k} g^{-s_k})^{1/(x_i - w)}, s_k)_{k \in \{3,4\}},$$

and it can test the validity of ciphertext by $C_5 = e(C_1, d_3^{\phi} d_4) C_2^{s_3 \phi + s_4}$.

Efficient but weaker scheme. If we add a new rule in game 2 of Definition 2, for example "(v). $\mathcal{A}$ cannot obtain the trapdoor query on $\langle pk_{i^*}, w^* \rangle$.", then we can prove the C-PRES-IND-ANON-CCA security of this efficient scheme. We call this scheme $\prod_1$. The reason why we need rule (v) in game 2 is due to the fact that the proxy with the trapdoor can modify the second level ciphertext. Actually, there are two kinds of CCA secure schemes: one is where the user with the private key cannot modify the ciphertext, the other is the user with the private key can modify the ciphertext. The main drawback of Gentry's IBE is clearly demonstrated in the situation where the user can use the ciphertext validity test key pair to modify the ciphertext to the new ciphertext without knowing the plaintext, and the new ciphertext can pass the validity test. Obviously, if we remove rule (v), the challenge ciphertext can be modified by the adversary who has the trapdoor on $\langle pk_{i^*}, w^* \rangle$. Further, the normal proxy cannot detect the modification.

Stronger scheme. As discussed above, we cannot use $H(C_1, C_2, C_3, C_4, C_5)^r$ to ensure the public verifiability of the second level ciphertext since it will jeopardize the keyword-anonymity. Therefore, we use strongly unforgeable one-time signatures by selecting a one-time signature key pair $(ssk, svk) \leftarrow \mathcal{G}(\lambda)$ and set $C_0 = svk$. Then, we generate a one-time signature $\sigma = \mathcal{S}(ssk, (C_1, C_2, C_3, C_4, C_5))$. Let $\phi = H_3(C_0, C_1, C_2, C_3, C_4)$ and $\phi' = H_0(C_0, C_2, C_4)$, thus the form of second level (original) ciphertext $(C_0, C_1, C_2, C_3, C_4, C_5)$ is

$$(svk, (X_i g^{-w})^r, e(g,g)^r, e(g, Y_{i,1})^{r\phi'} e(g, Y_{i,2})^r R, m \oplus H_2(R), e(g, Y_{i,3})^{r\phi} e(g, Y_{i,4})^r).$$

When the adversary modifies the ciphertext, it must change the svk, which entails that ϕ will be changed, and hence, it is impossible to modify C_5 without the trapdoor. Similarly, when ϕ' is changed, C_3 of the re-encrypted ciphertext will become a random value, then the decryption algorithm will output $\perp$ when decrypting this ciphertext.

The description of our anonymous conditional proxy re-encryption with keyword search scheme is as follows.

- *GlobalSetup(λ):* Let λ be the security parameter and $(p, g, \mathbb{G}_1, \mathbb{G}_2, e)$ be the bilinear map parameters. Let the message space be $\mathcal{M} = \{0,1\}^k$ and the condition space be $\mathcal{W} = \mathbb{Z}_p^*$. Let $H_0 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_1 : \{0,1\}^* \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_2 \to \{0,1\}^k$, and $H_3 : \{0,1\}^* \to \mathbb{Z}_p^*$ be three hash functions. Generate a strongly unforgeable one-time signature scheme $sig = (\mathcal{G}, \mathcal{S}, \mathcal{V})$. The global system parameters are $(p, g, \mathbb{G}_1, \mathbb{G}_2, e, k, H_0, H_1, H_2, H_3, sig)$.

- *KeyGen(i):* user i selects random $x_i, y_{i,1}, y_{i,2}, y_{i,3}, y_{i,4} \in \mathbb{Z}_p^*$, computes $X_i = g^{x_i}$, $Y_{i,1} = g^{y_{i,1}}$,$Y_{i,2} = g^{y_{i,2}}$, $Y_{i,3} = g^{y_{i,3}}$ and $Y_{i,4} = g^{y_{i,4}}$, sets his public key as $pk_i = (X_i, Y_{i,1}, Y_{i,2}, Y_{i,3}, Y_{i,4})$ and the secret key $sk_i = (x_i, y_{i,1}, y_{i,2}, y_{i,3}, y_{i,4})$.
- *RKeyGen(pk_i, sk_i, w, pk_j):* given user i's pubic key pk_i and secret key $sk_i = (x_i, y_{i,1}, y_{i,2}, y_{i,3}, y_{i,4})$, a condition w, and user j's public key $pk_j = (X_j, Y_{j,1}, Y_{j,1}, Y_{j,2}, Y_{j,3}, Y_{j,4})$, selects two random $s_1, s_2 \in \mathbb{Z}_p^*$, computes $d_k = (Y_{j,k} Y_{i,k}^{-1} g^{-s_k})^{1/(x_i - w)}$, sets the re-encryption key $rk_{i,w,j} = (d_k, s_k)_{k\in\{1,2\}}$.
- *Trapdoor(pk_i, sk_i, w):* given user i's pubic key pk_i and private key sk_i and a condition w, selects two random $s_3, s_4 \in \mathbb{Z}_p^*$, computes $d_k = (Y_{i,k} g^{-s_k})^{1/(x_i - w)}$, sets the trapdoor $T_{i,w} = (d_k, s_k)_{k\in\{3,4\}}$.
- *Enc1(pk_i, m):* To encrypt a message $m \in \mathcal{M}$ under the public key pk_i. Picks random $R \in \mathbb{G}_2^*$ and $svk \in \{0,1\}^{k_1}$, sets $C_0 = svk$ and $r = H_1(m, R)$, computes $C_2 = e(g,g)^r$, $C_4 = m \oplus H_2(R)$,

$$\phi' = H_0(C_0, C_2, C_4), \qquad C_3 = e(g, Y_{i,1})^{r\phi'} e(g, Y_{i,2})^r R$$

  outputs the first level ciphertext $CT_i = (C_0, C_2, C_3, C_4)$.
- *Enc2(pk_i, m, w):* To encrypt a message $m \in \mathcal{M}$ under the public key pk_i and condition $w \in \mathcal{W}$, do the following.

  (1) Selects a one-time signature key pair $(ssk, svk) \leftarrow \mathcal{G}(\lambda)$ and sets $C_0 = svk$.
  (2) Picks $R \in \mathbb{G}_2^*$, computes $r = H_1(m, R)$, and

$$\begin{aligned} &C_1 = (X_i g^{-w})^r, \qquad C_2 = e(g,g)^r, \qquad C_4 = m \oplus H_2(R) \\ &\phi' = H_0(C_0, C_2, C_4), \qquad C_3 = e(g, Y_{i,1})^{r\phi'} e(g, Y_{i,2})^r R \\ &\phi = H_3(C_0, C_1, C_2, C_3, C_4), \qquad C_5 = e(g, Y_{i,3})^{r\phi} e(g, Y_{i,4})^r. \end{aligned}$$

  (3) Generates a one-time signature $\sigma = \mathcal{S}(ssk, (C_1, C_2, C_3, C_4, C_5))$ on the pair $(C_1, C_2, C_3, C_4, C_5)$.
  (4) Then, outputs the second level ciphertext $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$.
- *Test(CT_i, $T_{i,w}$):* on input of a trapdoor $T_{i,w} = (d_k, s_k)_{k=\{3,4\}}$ and a second level ciphertext $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$, it first computes $\phi = H_3(C_0, C_1, C_2, C_3, C_4)$, tests if

$$\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1 \tag{1}$$

$$C_5 = e(C_1, d_3^{\phi} d_4) C_2^{s_3\phi + s_4}. \tag{2}$$

  If one of the checks fails, outputs "0", otherwise it outputs "1".
  To verify this, consider the following.

$$\begin{aligned} e(C_1, d_3^{\phi} d_4) C_2^{s_3\phi+s_4} &= e((X_i g^{-w})^r, ((Y_{i,3} g^{-s_3})^{1/(x_i-w)})^{\phi} ((Y_{i,4} g^{-s_4})^{1/(x_i-w)})) C_2^{s_3\phi+s_4} \\ &= e(g^{(x_i-w)r}, ((Y_{i,3} g^{-s_3})^{\phi} (Y_{i,4} g^{-s_4}))^{1/(x_i-w)}) C_2^{s_3\phi+s_4} \\ &= e(g^r, (Y_{i,3} g^{-s_3})^{\phi} (Y_{i,4} g^{-s_4})) C_2^{s_3\phi+s_4} \\ &= e(g^r, (Y_{i,3})^{\phi} (Y_{i,4})) e(g^r, (g^{-s_3})^{\phi} (g^{-s_4})) C_2^{s_3\phi+s_4} \\ &= e(g, Y_{i,3})^{r\phi} e(g, Y_{i,4})^r \\ &= C_5. \end{aligned}$$

- *ReEnc(CT_i, $rk_{i,w,j}$):* on input of a re-encryption key $rk_{i,w,j} = (d_k, s_k)_{k\in\{1,2\}}$ and a second level ciphertext $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$. Test if

$$\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1. \tag{3}$$

  If Eq. (3) holds, it computes $\phi' = H_0(C_0, C_2, C_4)$, CT_i, is re-encrypted by computing

$$C_3' = e(C_1, d_1^{\phi'} d_2) C_2^{s_1\phi'+s_2} \cdot C_3.$$

  The re-encrypted ciphertext (level 1) is $CT_j = (C_0, C_2, C_3', C_4)$.
  To verify this, consider the following.

$$\begin{aligned} C_3' &= e(C_1, d_1^{\phi'} d_2) C_2^{s_1\phi'+s_2} \cdot C_3 \\ &= e((X_i g^{-w})^r, ((Y_{j,1} Y_{i,1}^{-1} g^{-s_1})^{1/(x_i-w)})^{\phi'} ((Y_{j,2} Y_{i,2}^{-1} g^{-s_2})^{1/(x_i-w)})) C_2^{s_1\phi'+s_2} \cdot C_3 \\ &= e(g^{(x_i-w)r}, ((Y_{j,1} Y_{i,1}^{-1} g^{-s_1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1} g^{-s_2}))^{1/(x_i-w)}) C_2^{s_1\phi'+s_2} \cdot C_3 \\ &= e(g^r, ((Y_{j,1} Y_{i,1}^{-1} g^{-s_1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1} g^{-s_2})) C_2^{s_1\phi'+s_2} \cdot C_3 \\ &= e(g^r, ((Y_{j,1} Y_{i,1}^{-1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1})) e(g^r, (g^{-s_1})^{\phi'} (g^{-s_2})) (e(g,g)^r)^{s_1\phi'+s_2} \cdot C_3 \\ &= e(g, Y_{j,1})^{r\phi'} e(g, Y_{j,2})^r (e(g, Y_{i,1})^{r\phi'} e(g, Y_{i,2})^r)^{-1} e(g, Y_{i,1})^{r\phi'} e(g, Y_{i,2})^r R \\ &= e(g, Y_{j,1})^{r\phi'} e(g, Y_{j,2})^r R. \end{aligned}$$

- *Dec*1(CT_j, sk_j): On input a secret key sk_j and a first level ciphertext (re-encrypted ciphertext) $CT_j = (C_0, C_2, C_3, C_4)$, it computes $\phi' = H_0(C_0, C_2, C_4)$ and $R = C_3/(C_2)^{y_{j,1}\phi'+y_{j,2}}$, $m = C_4 \oplus H_2(R)$, $r = H_1(m, R)$, and checks whether $C_2 \stackrel{?}{=} e(g, g)^r$ holds. If yes, it returns m; else it returns $\perp$.
- *Dec*2(CT_i, sk_i): On input a secret key sk_i and a second level ciphertext $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$, it computes $\phi = H_3(C_0, C_1, C_2, C_3, C_4)$, $\phi' = H_0(C_0, C_2, C_4)$ and tests if

$$\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1 \tag{4}$$

$$C_5 = (C_2)^{y_{i,3}\phi+y_{i,4}}. \tag{5}$$

If one of the checks fails, outputs $\perp$. Otherwise, computes

$$R = C_3/(C_2)^{y_{i,1}\phi'+y_{i,2}}, \qquad m = C_4 \oplus H_2(R), \qquad r = H_1(m, R)$$

and checks whether $C_2 \stackrel{?}{=} e(g, g)^r$ holds. If yes, it returns m; else it returns $\perp$.

To verify this, consider the following.

$$\begin{aligned}(C_2)^{y_{i,3}\phi+y_{i,4}} &= (e(g, g)^r)^{y_{i,3}\phi+y_{i,4}} \\ &= C_5.\end{aligned}$$

$$\begin{aligned}C_3/(C_2)^{y_{i,1}\phi'+y_{i,2}} &= (e(g, Y_{i,1})^{r\phi'} e(g, Y_{i,2})^r R)/(e(g, g)^r)^{y_{i,1}\phi'+y_{i,2}} \\ &= R.\end{aligned}$$

3.2. Security of our C-PRES

In this subsection, we prove the C-PRES-IND-ANON-CCA security for our scheme in the random oracle model. The analysis of Game 1, Game 2 and Game 3 is as follows.

Theorem 1. *If the q-ABDHE and DBDH assumptions hold, and sig $= (\mathcal{G}, \mathcal{S}, \mathcal{V})$ is a strongly unforgeable one-time signature, then the scheme is C-PRES-IND-ANON-CCA secure in the random oracle model.*

Lemma 1. *If there exists an IND-ANON adversary $\mathcal{A}$ against our scheme, then there exists an algorithm $\mathcal{B}$ which can solve the q-ABDHE problem for $q \geq q_k + 1$, where q_k is the number of re-encryption key and trapdoor queries.*

Proof. Our approach to proving Lemma 1 closely follows the proof of security for Gentry's IBE scheme [18]. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, that can attack our scheme in the random oracle model. Let q_k be the number of re-encryption key queries. We build a simulator $\mathcal{B}$ that can play a q-ABDHE game for $q \geq q_k + 1$. In the following, we call *HU* the set of honest parties, and *CU* the set of corrupt parties. The simulation proceeds as follows:

We let the challenger set the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with an efficient bilinear map e and a generator g of $\mathbb{G}_1$. Simulator $\mathcal{B}$ inputs a q-ABDHE instance $(g, g^x, g^{x^2}, \ldots, g^{x^q}, g^z, g^{zx^{q+2}}, T)$, and has to distinguish $T = e(g, g)^{zx^{q+1}}$ from a random element in $\mathbb{G}_2$.

Before describing $\mathcal{B}$, we first define an event F_{OTS} and bound its probability to occur. Let $C^* = (svk^*, C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$ denote the challenge ciphertext given to $\mathcal{A}$ in the game. Let F_{OTS} be the event that $\mathcal{A}$ issues a test or re-encryption query for ciphertext $C^* = (svk^*, C_1, C_2, C_3, C_4, C_5, \sigma)$ but $\mathcal{V}(svk^*, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1$. In the "phase 1" stage, $\mathcal{A}$ has simply no information on svk^*. Hence, the probability of a pre-challenge occurrence of F_{OTS} does not exceed $q_t\theta$ if q_t is the overall number of re-encryption and test oracle queries and θ denotes the maximal probability (which by assumption does not exceed $1/p$) that any one-time verification key svk^* is output by $\mathcal{G}$. In the "phase 2" stage, F_{OTS} clearly gives rise to an algorithm breaking the strong unforgeability of the one-time signature. Therefore, the probability $Pr[F_{OTS}] \leq q_t/p + Adv^{OTS}$, where the second term accounts for the fact that the probability of definition one time signature, must be negligible by assumption.

The random oracles H_0, H_1, H_2, and H_3 are controlled by $\mathcal{B}$ as follows.

If $\mathcal{A}$ queries (C_0, C_2, C_4) to the random oracle H_0, $\mathcal{B}$ searches H_0^{List} for an entry (C_0, C_2, C_4, ψ'). If it exists, return ψ' as answer. Otherwise, it chooses $\psi' \in \mathbb{Z}_p^*$ at random and returns it as the answer and places (C_0, C_2, C_4, ψ') into H_0^{List}.

If $\mathcal{A}$ queries (m, R) to the random oracle H_1, $\mathcal{B}$ searches H_1^{List} for an entry (m, R, r). If it exists, return r as answer. Otherwise, it chooses $r \in \mathbb{Z}_p^*$ at random and returns it as the answer and places (m, R, r) into H_1^{List}.

If $\mathcal{A}$ queries (R) to the random oracle H_2, $\mathcal{B}$ searches H_2^{List} for an entry (R, ω). If it exists, return ω as answer. Otherwise, it chooses $\omega \in \{0, 1\}^k$ at random and returns it as the answer and places (R, ω) into H_2^{List}.

If $\mathcal{A}$ queries $(C_0, C_1, C_2, C_3, C_4)$ to the random oracle H_3, $\mathcal{B}$ searches H_3^{List} for an entry $(C_0, C_1, C_2, C_3, C_4, \psi)$. If it exists, return ψ as answer. Otherwise, it chooses $\psi \in \mathbb{Z}_p^*$ at random and returns it as the answer and places $(C_0, C_1, C_2, C_3, C_4, \psi)$ into H_3^{List}.

(1) Setup: Let λ be the security parameter and $(p, g, \mathbb{G}_1, \mathbb{G}_2, e)$ be the bilinear map parameters. Let message space be $\mathcal{M} = \{0, 1\}^k$ and condition space be $\mathcal{W} = \mathbb{Z}_p^*$. The global system parameters are $(p, g, \mathbb{G}_1, \mathbb{G}_2, e, k, H_0, H_1, H_2, H_3, sig)$.

(2) Query phase 1. $\mathcal{A}$ makes the following queries:

- Uncorrupted key generation query $\langle i\rangle$: public keys of honest user $i \in HU$ are defined as follows: $\mathcal{B}$ selects a random value $\eta_i \in \mathbb{Z}_p^*$, computes $X_i = g^{x+\eta_i}$, $\mathcal{B}$ picks four random degree q polynomials $f_{i,k}(X)$ where $k \in \{1, 2, 3, 4\}$, and defines $\{Y_{i,k} = g^{f_{i,k}(x)}\}_{k\in\{1,2,3,4\}}$. This implicitly defines the secret key value as $x_i = x + \eta_i$, $\{y_{i,k} = f_{i,k}(x)\}_{k\in\{1,2,3,4\}}$. $\mathcal{B}$ sets target user's public key as $pk_{i^*} = (X_i, \{Y_{i,k}\}_{k\in\{1,2,3,4\}})$, and sends public key to $\mathcal{A}$.
- Corrupted key generation query $\langle i\rangle$: Public keys of corrupt user $i \in CU$ are the same as the key generation algorithm, this means the simulator $\mathcal{B}$ can know both the public key and secret key of user $i \in CU$, and then sends (pk_i, sk_i) to $\mathcal{A}$.
- Re-encryption key query $\langle pk_i, w, pk_j\rangle$: $\mathcal{B}$ has to distinguish several situations:
  (a) If $i \in CU$, since $\mathcal{B}$ can know the secret key part x_i for user i, so $\mathcal{B}$ can compute it correctly.
  (b) If $i \in HU$ and $j \in CU$, let $F_{i,j,k}(X) = y_{j,k} - f_{i,k}(X)$, computes

$$s_{w,k} = F_{i,j,k}(w - \eta_i), \qquad d_{w,k} = g^{(F_{i,j,k}(x) - F_{i,j,k}(w-\eta_i))/(x+\eta_i-w)}.$$

  (c) If $i \in HU$ and $j \in HU$, let $F_{i,j,k}(X) = f_{j,k}(X) - f_{i,k}(X)$, then $\mathcal{B}$ computes

$$s_{w,k} = F_{i,j,k}(w - \eta_i), \qquad d_{w,k} = g^{(F_{i,j,k}(x) - F_{i,j,k}(w-\eta_i))/(x+\eta_i-w)}.$$

  Clearly, it is easy to compute $d_{w,k}$ using the Lagrange form of Taylor's theorem [2]. When $q \geq q_k + 1$, $f_{i,k}(w - \eta_i), f_{j,k}(w - \eta_i)$ are random values from $\mathcal{A}$'s view, since $f_{i,k}(X)$ and $f_{j,k}(X)$ are random degree q polynomials.
  Sends the re-encryption key $rk_{i,w,j} = \{d_{w,k}, s_{w,k}\}_{k\in\{1,2\}}$ to $\mathcal{A}$.
- Trapdoor query $\langle pk_i, w\rangle$: $\mathcal{B}$ has to distinguish several situations:
  (a) If $i \in CU$, since $\mathcal{B}$ can know the secret key part x_i for user i, so $\mathcal{B}$ can compute it correctly.
  (b) If $i \in HU$, then $\mathcal{B}$ computes

$$\{s_{w,k} = f_{i,k}(w - \eta_i)\}_{k\in\{3,4\}}, \qquad d_{w,k} = g^{(f_{i,k}(x) - f_{i,k}(w-\eta_i))/(x+\eta_i-w)}.$$

  When $q \geq q_k + 1$, $\{f_{i,k}(w - \eta_i)\}_{k\in\{3,4\}}$ are random values from $\mathcal{A}$'s view, since $f_{i,k}(X)$ where $k \in \{3, 4\}$ are random degree q polynomials.
  Sends the trapdoor $T_{i,w} = \{d_{w,k}, s_{w,k}\}_{k\in\{3,4\}}$ to $\mathcal{A}$.
- Test query $\langle pk_i, w, CT_i\rangle$: $\mathcal{A}$ can adaptively ask $\mathcal{B}$ for the test query for public key pk_i, any keyword w and any ciphertext of his choice. If F_{OTS} occurs, the process halts (an occurrence of F_{OTS} in phase 1 is different from that in phase 2 as discussed in the preparation phase). If F_{OTS} does not occur, $\mathcal{B}$ first queries a trapdoor query on $\langle pk_i, w\rangle$ to get the trapdoor $T_{i,w}$ and then responds the result $Test(CT_i, T_{i,w})$ to $\mathcal{A}$.
- Re-encryption query $\langle pk_i, pk_j, w, CT_i\rangle$: If F_{OTS} occurs, the process halts. If F_{OTS} does not occur, since $\mathcal{B}$ can compute unidirectional re-encryption key $rk_{i,w,j}$ for all user i and j, so $\mathcal{B}$ can compute it correctly.
- Decryption query $\langle pk_j, CT_j\rangle$: If $\langle pk_j, CT_j\rangle$ denotes the queries on re-encrypted ciphertext (first level ciphertext), $CT_j = (C_0, C_2, C_3, C_4)$. For a user's $j \in CU$, $\mathcal{B}$ can decrypt it correctly, since $\mathcal{B}$ knows the secret key for user $j \in CU$. For a user's $j \in HU$, then $\mathcal{B}$ searches H_0^{List}, H_1^{List} and H_2^{List} to see whether there exist a tuple (C_0, C_2, C_4, ϕ'), a tuple (m, R, r) and a tuple (R, ω) such that

$$C_2 = e(g, g)^r, \qquad C_3 = e(g, Y_{j,1})^{r\phi'} e(g, Y_{j,2})^r R, \qquad C_4 = m \oplus H_2(R).$$

  If yes, it outputs m to $\mathcal{A}$; else outputs $\perp$.
- Decryption query $\langle pk_i, w, CT_i\rangle$: If $\langle pk_i, w, CT_i\rangle$ denotes the queries on second level ciphertext CT_i. $\mathcal{B}$ makes a re-encryption query on $\langle pk_i, pk_j, w, CT_i\rangle$ to get the re-encrypted ciphertext (first level ciphertext) CT_j, then makes a decryption query on $\langle pk_j, CT_j\rangle$, and sends the result to $\mathcal{A}$.

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, a condition pair (w_0, w_1) and a plaintext m. $\mathcal{B}$ responds by choosing a random $\beta \in \{0, 1\}$, sets $\{s^*_{w_\beta,k} = f_{i^*,k}(w_\beta - \eta_{i^*})\}_{k\in\{1,2,3,4\}}$, then $\mathcal{B}$ computes

$$d^*_{w_\beta,k} = g^{(f_{i^*,k}(x) - f_{i^*,k}(w_\beta - \eta_{i^*}))/(x+\eta_{i^*}-w_\beta)}$$

where $k \in \{1, 2, 3, 4\}$. $\mathcal{B}$ also selects a strongly unforgeable one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$, and sets $C_0^* = svk^*$.

Define the degree $q + 1$ polynomial

$$F^*(X) = (X^{q+2} - (w_\beta - \eta_{i^*})^{q+2})/(X + \eta_{i^*} - w_\beta) = \sum_{i=0}^{q+1}(F_i^* X^i).$$

Picks random $R^* \in \mathbb{G}_2^*$ and computes

$$C_1^* = g^{zx^{q+2}}(g^z)^{-(w_\beta - \eta_{i^*})^{q+2}}$$

$$C_2^* = T^{F^*_{q+1}} e\left(g^z, \prod_{i=0}^{q}(g^{x^i})^{F_i^*}\right)$$

$$C_4^* = m \oplus H_2(R^*)$$
$$\phi'^* = H_0(C_0^*, C_2^*, C_4^*)$$
$$C_3^* = R^* \cdot e(C_1^*, (d_{w_\beta,1}^*)^{\phi'^*} d_{w_\beta,2}^*) \cdot (C_2^*)^{s_{w_\beta,1}^* \phi'^* + s_{w_\beta,2}^*}$$
$$\phi^* = H_3(C_0^*, C_1^*, C_2^*, C_3^*, C_4^*)$$
$$C_5^* = e(C_1^*, (d_{w_\beta,3}^*)^{\phi^*} d_{w_\beta,4}^*) \cdot (C_2^*)^{s_{w_\beta,3}^* \phi^* + s_{w_\beta,4}^*}.$$

Generates a strongly unforgeable one-time signature $\sigma^* = \mathcal{S}(ssk^*, (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*))$. Sends the challenge ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$ to $\mathcal{A}$.

To see this, implicitly let $H_1(m, R^*) = r^* = zF^*(x)$, if $T = e(g,g)^{zx^{q+1}}$, then $C_1^* = g^{(x+\eta_{i^*}-w_\beta)r^*} = (X_{i^*} g^{-w_\beta})^{r^*}$, $C_2^* = e(g,g)^{r^*}$, $C_3^* = R^* \cdot e(g, Y_{i^*,1})^{\phi'^* r^*} e(g, Y_{i^*,2})^{r^*}$ and $C_5^* = e(g, Y_{i^*,3})^{\phi^* r^*} e(g, Y_{i^*,4})^{r^*}$.

(4) Query phase 2. $\mathcal{A}$ continues making queries as in the query phase 1.

(5) Guess. $\mathcal{A}$ outputs the guess β', if $\beta' = \beta$, then output 1 meaning $T = e(g,g)^{zx^{q+1}}$; else output 0 meaning $T = e(g,g)^r$.

Probability analysis: When F_{OTS} does not occur, if $T = e(g,g)^{zx^{q+1}}$, then the simulation is perfect, and $\mathcal{A}$ will guess the bit β correctly with probability $1/2 + \epsilon$. Else, T is uniformly random, and thus (C_1^*, C_2^*) is a uniformly random and independent element. In this case, the inequality $C_2^* \neq e(C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)}$ holds with probability $1 - 1/p$. When this inequality holds, the value of

$$K^* = e(C_1^*, (d_{w_\beta,1}^*)^{\phi'^*} d_{w_\beta,2}^*) \cdot (C_2^*)^{s_{w_\beta,1}^* \phi'^* + s_{w_\beta,2}^*} \tag{6}$$
$$= e(C_1^*, (Y_{i^*,1}^{\phi'^*} Y_{i^*,2})^{1/(x+\eta_{i^*}-w_\beta)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)})} \right)^{s_{w_\beta,1}^* \phi'^* + s_{w_\beta,2}^*} \tag{7}$$

is uniformly random and independent from $\mathcal{A}$'s view (except for the value C_3^*), since $s_{w_\beta,k}^*$ (when $q \geq q_k + 1$, $\{s_{w_\beta,k}^* = f_{i,k}(w_\beta - \eta_{i^*})\}$ is a random value from $\mathcal{A}$'s view). Thus, C_3^* is uniformly random and independent, and $(C_1^*, C_2^*, C_3^*, C_4^*)$ can reveal no information regarding the random bit β. Similarly,

$$C_5^* = e(C_1^*, (Y_{i^*,3}^{\phi^*} Y_{i^*,4})^{1/(x+\eta_{i^*}-w_\beta)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)})} \right)^{s_{w_\beta,3}^* \phi^* + s_{w_\beta,4}^*} \tag{8}$$

thus C_5^* will not leak the bit β.

Finally, we still need to explain why the re-encryption and test queries do not jeopardize the security of the scheme. For the re-encryption query, we need to prove the re-encryption query on $\langle pk_i, pk_j, w, CT_i \rangle$ where $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ and $\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1$ is a mauling of the challenge ciphertext may not leak the bit β. Querying valid ciphertexts where $C_2 = e(C_1, g)^{1/(x+\eta_i - w)}$ to re-encryption oracle does not help $\mathcal{A}$ distinguish between the simulation and the actual construction, because of the correctness of re-encryption. For invalid ciphertexts where $C_2 \neq e(C_1, g)^{1/(x+\eta_i-w)}$, the ciphertext will be re-encrypted by

$$C_3' = C_3 \cdot e(C_1, d_{w,1}^{\phi'} d_{w,2}) C_2^{s_{w,1}\phi' + s_{w,2}}$$
$$= C_3 \cdot e(C_1, ((Y_{j,1} Y_{i,1}^{-1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1}))^{1/(x+\eta_i - w)}) \left(\frac{C_2}{e((C_1, g)^{1/(x+\eta_i-w)})} \right)^{s_{w,1}\phi' + s_{w,2}}.$$

When $C_3 \neq C_3^*$, since $s_{w,k}$ is uniformly random and independent from $\mathcal{A}$'s view, then the re-encryption result C_3' is a random value from $\mathcal{A}$'s view. Thus it will not leak the bit β.

When $C_3 = C_3^*$, we have

$$C_3' = R^* e(C_1^*, (Y_{i^*,1}^{\phi'^*} Y_{i^*,2})^{1/(x+\eta_{i^*}-w_\beta)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)})} \right)^{s_{w_\beta,1}^* \phi'^* + s_{w_\beta,2}^*}$$
$$\cdot e(C_1, ((Y_{j,1} Y_{i,1}^{-1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1}))^{1/(x+\eta_i - w)}) \left(\frac{C_2}{e((C_1, g)^{1/(x+\eta_i-w)})} \right)^{s_{w,1}\phi' + s_{w,2}}.$$

If $C_1 \neq C_1^*$ or $C_2 \neq C_2^*$ or $w \neq w_\beta$ or $\eta_i \neq \eta_{i^*}$, since $s_{w_\beta,k}^*$ is uniformly random and independent from $\mathcal{A}$'s view, then the re-encryption result C_3' is a random value from $\mathcal{A}$'s view. Thus it will not leak the bit β.

If $C_1 = C_1^*, C_2 = C_2^*, w = w_\beta, \eta_i = \eta_{i^*}$ and $C_3 = C_3^*$, this situation can be discussed in two cases:

- If $C_0 = svk \equiv svk^* = C_0^*$, then

  $(C_1, C_2, C_3, C_4, C_5, \sigma) \neq (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$

  $\mathcal{B}$ is faced with an occurrence of F_{OTS} and halts.

- If $C_0 = svk \neq svk^* = C_0^*$, we have $\phi \neq \phi^*$ and $\phi' \neq \phi'^*$,

$$C_3' = R^* e(C_1^*, (Y_{i^*,1}^{\phi'^*} Y_{i^*,2})^{1/(x+\eta_{i^*}-w_\beta)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)})}\right)^{s^*_{w_\beta,1}\phi'^* + s^*_{w_\beta,2}}$$
$$\cdot e(C_1, ((Y_{j,1}Y_{i,1}^{-1})^{\phi'}(Y_{j,2}Y_{i,2}^{-1}))^{1/(x+\eta_i-w)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)})}\right)^{s^*_{w_\beta,1}\phi' + s^*_{w_\beta,2}}.$$

Since $\phi' \neq \phi'^*$, then $l = s^*_{w_\beta,1}\phi' + s^*_{w_\beta,2}$ is uniformly random and independent from $l^* = s^*_{w_\beta,1}\phi'^* + s^*_{w_\beta,2}$. Then the re-encryption result C_3' is a random value from $\mathcal{A}$'s view. Thus it will not leak the bit β.

Similarly, we need to prove the test query on $\langle pk_i, w, CT_i\rangle$ where $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ and $\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1$ is a mauling of the challenge ciphertext may not leak the bit β. If $\langle pk_i, w\rangle \neq \langle pk_{i^*}, w^*\rangle$, test querying on this will not help $\mathcal{A}$ since $\mathcal{A}$ can make the trapdoor query on $\langle pk_i, w\rangle$. Querying valid ciphertexts where $C_2 = e(C_1, g)^{1/(x+\eta_{i^*}-w^*)}$ to test oracle does not help $\mathcal{A}$ to distinguish between the simulation and the actual construction, due to the correctness of test. For invalid ciphertexts where $C_2 \neq e(C_1, g)^{1/(x+\eta_{i^*}-w^*)}$, the ciphertext will be verified by the following equation.

$$\begin{aligned} C_5 &= e(C_1, d^*_{w^*,3}{}^{\phi'} d^*_{w^*,4}) C_2^{s^*_{w^*,3}\phi' + s^*_{w^*,4}} \\ &= e(C_1, ((Y_{i^*,3}g^{-s^*_{w^*,3}})^{1/(x_{i^*}-w^*)})^{\phi'}((Y_{i^*,4}g^{-s^*_{w^*,4}})^{1/(x_{i^*}-w^*)})) C_2^{s^*_{w^*,3}\phi' + s^*_{w^*,4}} \\ &= e(C_1, (Y_{i^*,3}^{\phi'} Y_{i^*,4})^{1/(x+\eta_{i^*}-w^*)}) \left(\frac{C_2}{e((C_1, g)^{1/(x+\eta_{i^*}-w^*)})}\right)^{s^*_{w^*,3}\phi' + s^*_{w^*,4}}. \end{aligned}$$

Similarly from the challenge ciphertext, we have

$$\begin{aligned} C_5^* &= e(C_1^*, d^*_{w_\beta,3}{}^{\phi'} d^*_{w_\beta,4}) C_2^{*\, s^*_{w_\beta,3}\phi' + s^*_{w_\beta,4}} \\ &= e(C_1^*, ((Y_{i^*,3}g^{-s^*_{w_\beta,3}})^{1/(x_{i^*}-w_\beta)})^{\phi'}((Y_{i^*,4}g^{-s^*_{w_\beta,4}})^{1/(x_{i^*}-w_\beta)}))(C_2^*)^{s^*_{w_\beta,3}\phi' + s^*_{w_\beta,4}} \\ &= e(C_1^*, (Y_{i^*,3}^{\phi'} Y_{i^*,4})^{1/(x+\eta_{i^*}-w_\beta)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w_\beta)})}\right)^{s^*_{w_\beta,3}\phi' + s^*_{w_\beta,4}}. \end{aligned}$$

This scenario can be divided into two cases:

- If $C_0 = svk \equiv svk^* = C_0^*$, then

  $(C_1, C_2, C_3, C_4, C_5, \sigma) \neq (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$

  $\mathcal{B}$ is faced with an occurrence of F_{OTS} and halts.
- If $C_0 = svk \neq svk^* = C_0^*$, we have $\phi' \neq \phi'^*$. If $w^* \neq w_\beta$, since $s^*_{w^*,3}$ and $s^*_{w^*,4}$ are uniformly random and independent from $\mathcal{A}$'s view, then it will not leak the bit β. If $w^* = w_\beta$, then $l = s^*_{w_\beta,3}\phi' + s^*_{w_\beta,4}$ is uniformly random and independent from $l^* = s^*_{w_\beta,3}\phi'^* + s^*_{w_\beta,4}$. Then the test result will not leak the bit β, since $s^*_{w_\beta,3}$ and $s^*_{w_\beta,4}$ are uniformly random and independent from $\mathcal{A}$'s view. Actually, the test query will always output "0" since the valid C_5 which is hidden by $l = s^*_{w_\beta,3}\phi' + s^*_{w_\beta,4}$ is a random value from $\mathcal{A}$'s view.

This completes the proof of Lemma 1. □

Lemma 2. *If there exists an IND-L2-CCA adversary $\mathcal{A}$ against our scheme, then there exists an algorithm $\mathcal{B}$ which can solve the q-ABDHE problem for $q \geq q_k + 1$, where q_k is the number of re-encryption key and trapdoor queries.*

Proof. Our approach to proving Lemma 2 closely follows the proof of Lemma 1. Suppose there exists a polynomial-time adversary, $\mathcal{A}$, that can attack our scheme in the random oracle model. Let q_k is the number of re-encryption key queries. We build a simulator $\mathcal{B}$ that can play a q-ABDHE game for $q \geq q_k + 1$. In the following, we call *HU* the set of honest parties, and *CU* the set of corrupt parties. The simulation proceeds as follows:

We let the challenger set the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with an efficient bilinear map e and a generator g of $\mathbb{G}_1$. Simulator $\mathcal{B}$ inputs a q-ABDHE instance $(g, g^x, g^{x^2}, \ldots, g^{x^q}, g^z, g^{zx^{q+2}}, T)$, and has to distinguish $T = e(g, g)^{zx^{q+1}}$ from a random element in $\mathbb{G}_2$.

The random oracles H_0, H_1, H_2, and H_3 controlled by $\mathcal{B}$ are the same as in Lemma 1.

(1) Setup: Let λ be the security parameter and $(p, g, \mathbb{G}_1, \mathbb{G}_2, e)$ be the bilinear map parameters. Let message space be $\mathcal{M} = \{0, 1\}^k$ and condition space be $\mathcal{W} = \mathbb{Z}_p^*$. The global system parameters are $(p, g, \mathbb{G}_1, \mathbb{G}_2, e, k, H_0, H_1, H_2, H_3, sig)$.

(2) Query phase 1. $\mathcal{A}$ makes the following queries:

- Uncorrupted key generation query $\langle i\rangle$: public keys of honest user $i \in HU$ are defined as the following: $\mathcal{B}$ selects a random value $\eta_i \in \mathbb{Z}_p^*$, computes $X_i = g^{x+\eta_i}$, $\mathcal{B}$ picks four random degree q polynomials $f_{i,k}(X)$ where $k \in \{1, 2, 3, 4\}$, and defines $\{Y_{i,k} = g^{f_{i,k}(x)}\}_{k\in\{1,2,3,4\}}$. This implicitly defines the secret key value as $x_i = x + \eta_i$, $\{y_{i,k} = f_{i,k}(x)\}_{k\in\{1,2,3,4\}}$. $\mathcal{B}$ sets target user's public key as $pk_{i^*} = (X_i, \{Y_{i,k}\}_{k\in\{1,2,3,4\}})$, and sends public key to $\mathcal{A}$.
- Corrupted key generation query $\langle i\rangle$: Public keys of corrupt user $i \in CU$ are the same as the key generation algorithm, this means the simulator $\mathcal{B}$ can know the both the public key and secret key of user $i \in CU$, and then sends (pk_i, sk_i) to $\mathcal{A}$.
- Re-encryption key query $\langle pk_i, w, pk_j\rangle$: $\mathcal{B}$ has to distinguish several situations:
  (a) If $i \in CU$, since $\mathcal{B}$ can know the secret key part x_i for user i, so $\mathcal{B}$ can compute it correctly.
  (b) If $i \in HU$ and $j \in CU$, let $F_{i,j,k}(X) = y_{j,k} - f_{i,k}(X)$, computes

$$s_{w,k} = F_{i,j,k}(w - \eta_i), \qquad d_{w,k} = g^{(F_{i,j,k}(x) - F_{i,j,k}(w-\eta_i))/(x+\eta_i-w)}.$$

  (c) If $i \in HU$ and $j \in HU$, let $F_{i,j,k}(X) = f_{j,k}(X) - f_{i,k}(X)$, then $\mathcal{B}$ computes

$$s_{w,k} = F_{i,j,k}(w - \eta_i), \qquad d_{w,k} = g^{(F_{i,j,k}(x) - F_{i,j,k}(w-\eta_i))/(x+\eta_i-w)}.$$

  When $q \geq q_k + 1$, $f_{i,k}(w - \eta_i)$, $f_{j,k}(w - \eta_i)$ are random values from $\mathcal{A}$'s view, since $f_{i,k}(X)$ and $f_{j,k}(X)$ are random degree q polynomials.

  Sends the re-encryption key $rk_{i,w,j} = \{d_{w,k}, s_{w,k}\}_{k\in\{1,2\}}$ to $\mathcal{A}$.
- Trapdoor query $\langle pk_i, w\rangle$: $\mathcal{B}$ has to distinguish several situations:
  (a) If $i \in CU$, since $\mathcal{B}$ can know the secret key part x_i for user i, so $\mathcal{B}$ can compute it correctly.
  (b) If $i \in HU$, then $\mathcal{B}$ computes

$$\{s_{w,k} = f_{i,k}(w - \eta_i)\}_{k\in\{3,4\}}, \qquad d_{w,k} = g^{(f_{i,k}(x) - f_{i,k}(w-\eta_i))/(x+\eta_i-w)}.$$

  When $q \geq q_k + 1$, $\{f_{i,k}(w - \eta_i)\}_{k\in\{3,4\}}$ are random values from $\mathcal{A}$'s view, since $f_{i,k}(X)$ where $k \in \{3, 4\}$ are random degree q polynomials.

  Sends the trapdoor $T_{i,w} = \{d_{w,k}, s_{w,k}\}_{k\in\{3,4\}}$ to $\mathcal{A}$.
- Test query $\langle pk_i, w, CT_i\rangle$: $\mathcal{A}$ can adaptively ask $\mathcal{B}$ for the test query for public key pk_i, any keyword w and any ciphertext of his choice. $\mathcal{B}$ first query a trapdoor query on $\langle pk_i, w\rangle$ to get the trapdoor $T_{i,w}$ and then responds the result $Test(CT_i, T_{i,w})$ to $\mathcal{A}$.
- Re-encryption query $\langle pk_i, pk_j, w, CT_i\rangle$: Since $\mathcal{B}$ can compute unidirectional re-encryption key $rk_{i,w,j}$ and the trapdoor $T_{i,w} = \{d_{w,k}, s_{w,k}\}_{k\in\{1,2\}}$ for all user i and j, so $\mathcal{B}$ can compute it correctly.
- Decryption query $\langle pk_j, CT_j\rangle$: If $\langle pk_j, CT_j\rangle$ denotes the queries on re-encrypted ciphertext (first level ciphertext), $CT_j = (C_0, C_2, C_3, C_4)$. For a user's $j \in CU$, $\mathcal{B}$ can decrypt it correctly, since $\mathcal{B}$ knows the secret key for user $j \in CU$. For a user's $j \in HU$, then $\mathcal{B}$ searches for H_0^{List}, H_1^{List} and H_2^{List} to see whether there exists a tuple (C_0, C_2, C_4, ϕ'), a tuple (m, R, r) and a tuple (R, ω) such that

$$C_2 = e(g, g)^r, \qquad C_3 = e(g, Y_{j,1})^{r\phi'} e(g, Y_{j,2})^r R, \qquad C_4 = m \oplus H_2(R).$$

  If yes, it outputs m to $\mathcal{A}$; else it outputs $\perp$.
- Decryption query $\langle pk_i, w, CT_i\rangle$: If $\langle pk_i, w, CT_i\rangle$ denotes the queries on second level ciphertext CT_i. $\mathcal{B}$ makes a re-encryption query on $\langle pk_i, pk_j, w, CT_i\rangle$ to get the re-encrypted ciphertext (first level ciphertext) CT_j, then makes a decryption query on $\langle pk_j, CT_j\rangle$, and sends the result to $\mathcal{A}$.

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, a condition pair w^* and two equal length plaintexts (m_0, m_1). $\mathcal{B}$ responds by choosing a random $\beta \in \{0, 1\}$, sets $\{s^*_{w^*,k} = f_{i^*,k}(w^* - \eta_{i^*})\}_{k\in\{1,2,3,4\}}$, then $\mathcal{B}$ computes

$$d^*_{w^*,k} = g^{(f_{i^*,k}(x) - f_{i^*,k}(w^* - \eta_{i^*}))/(x+\eta_{i^*}-w^*)}$$

where $k \in \{1, 2, 3, 4\}$. $\mathcal{B}$ also selects a strongly unforgeable one-time signature key pair $(ssk^*, svk^*) \leftarrow \mathcal{G}(\lambda)$, sets $C_0^* = svk^*$.

Define the degree $q + 1$ polynomial

$$F^*(X) = (X^{q+2} - (w^* - \eta_{i^*})^{q+2})/(X + \eta_{i^*} - w^*) = \sum_{i=0}^{q+1} (F_i^* X^i).$$

Picks random $R^* \in \mathbb{G}_2^*$ and computes

$$C_1^* = g^{zx^{q+2}} (g^z)^{-(w^* - \eta_{i^*})^{q+2}}$$

$$C_2^* = T^{F^*_{q+1}} e\left(g^z, \prod_{i=0}^{q} (g^{x^i})^{F_i^*}\right)$$

$$C_4^* = m_\beta \oplus H_2(R^*)$$
$$\phi'^* = H_0(C_0^*, C_2^*, C_4^*)$$
$$C_3^* = R^* \cdot e(C_1^*, (d_{w^*,1}^*)^{\phi'^*} d_{w^*,2}^*) \cdot (C_2^*)^{s_{w^*,1}^* \phi'^* + s_{w^*,2}^*}$$
$$\phi^* = H_3(C_0^*, C_1^*, C_2^*, C_3^*, C_4^*)$$
$$C_5^* = e(C_1^*, (d_{w^*,3}^*)^{\phi^*} d_{w^*,4}^*) \cdot (C_2^*)^{s_{w^*,3}^* \phi^* + s_{w^*,4}^*}.$$

Generates a strongly unforgeable one-time signature $\sigma^* = \mathcal{S}(ssk^*, (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*))$. Sends the challenge ciphertext $C^* = (C_0^*, C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$ to $\mathcal{A}$.

To see this, implicitly let $H_1(m_\beta, R^*) = r^* = zF^*(x)$, if $T = e(g,g)^{zx^{q+1}}$, then $C_1^* = g^{(x+\eta_{i^*}-w^*)r^*} = (X_{i^*} g^{-w^*})^{r^*}$, $C_2^* = e(g,g)^{r^*}$, $C_3^* = R^* \cdot e(g, Y_{i^*,1})^{\phi'^* r^*} e(g, Y_{i^*,2})^{r^*}$ and $C_5^* = e(g, Y_{i^*,3})^{\phi^* r^*} e(g, Y_{i^*,4})^{r^*}$.

(4) Query phase 2. $\mathcal{A}$ continues making queries as in the query phase 1.

(5) Guess. $\mathcal{A}$ outputs the guess β', if $\beta' = \beta$, then output 1 meaning $T = e(g,g)^{zx^{q+1}}$; else output 0 meaning $T = e(g,g)^r$.

Probability Analysis: Similar to the analysis in Game 1, when F_{OTS} does not occur, if $T = e(g,g)^{zx^{q+1}}$, then the simulation is perfect, and $\mathcal{A}$ will guess the bit β correctly with probability $1/2 + \epsilon$. Else, T is uniformly random, and thus (C_1^*, C_2^*) is a uniformly random and independent element. In this case, the inequality $C_2^* \neq e(C_1^*, g)^{1/(x+\eta_{i^*}-w^*)}$ holds with probability $1 - 1/p$. When the inequality holds, the value of

$$K^* = e(C_1^*, (d_{w^*,1}^*)^{\phi'^*} d_{w^*,2}^*) \cdot (C_2^*)^{s_{w^*,1}^* \phi'^* + s_{w^*,2}^*} \tag{9}$$
$$= e(C_1^*, (Y_{i^*,1}^{\phi'^*} Y_{i^*,2})^{1/(x+\eta_{i^*}-w^*)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w^*)})} \right)^{s_{w^*,1}^* \phi'^* + s_{w^*,2}^*} \tag{10}$$

is uniformly random and independent from $\mathcal{A}$'s view (except for the value C_3^*), since $s_{w^*,k}^*$ (when $q \geq q_k + 1$, $\{s_{w^*,k}^* = f_{i,k}(w^* - \eta_{i^*})\}$ is a random value from $\mathcal{A}$'s view). Thus, C_3^* is uniformly random and independent, and (C_1^*, C_2^*, C_3^*) can reveal no information regarding the random value R^*. Thus, C_4^* will not leak the bit β. Similarly,

$$C_5^* = e(C_1^*, (Y_{i^*,3}^{\phi^*} Y_{i^*,4})^{1/(x+\eta_{i^*}-w^*)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w^*)})} \right)^{s_{w^*,3}^* \phi^* + s_{w^*,4}^*}$$

thus C_5^* will not leak the bit β. Finally, we need to prove the re-encryption query on $\langle pk_i, pk_j, w, CT_i \rangle$ where $CT_i = (C_0, C_1, C_2, C_3, C_4, C_5, \sigma)$ and $\mathcal{V}(C_0, \sigma, (C_1, C_2, C_3, C_4, C_5)) = 1$ is a mauling of the challenge ciphertext may not leak the bit β. Querying valid ciphertexts where $C_2 = e(C_1, g)^{1/(x+\eta_i - w)}$ to re-encryption oracle does not help $\mathcal{A}$ distinguish between the simulation and the actual construction, because of the correctness of the re-encryption. For invalid ciphertexts where $C_2 \neq e(C_1, g)^{1/(x+\eta_i-w)}$, the ciphertext will be re-encrypted by

$$C_3' = C_3 \cdot e(C_1, d_{w,1}^{\phi'} d_{w,2}) C_2^{s_{w,1}\phi' + s_{w,2}}$$
$$= C_3 \cdot e(C_1, ((Y_{j,1} Y_{i,1}^{-1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1}))^{1/(x+\eta_i - w)}) \left(\frac{C_2}{e((C_1, g)^{1/(x+\eta_i-w)})} \right)^{s_{w,1}\phi' + s_{w,2}}.$$

When $C_3 \neq C_3^*$, since $s_{w,k}$ is uniformly random and independent from $\mathcal{A}$'s view, then the re-encryption result C_3' is a random value from $\mathcal{A}$'s view. Thus it will not leak the bit β.

When $C_3 = C_3^*$, we have

$$C_3' = R^* e(C_1^*, (Y_{i^*,1}^{\phi'^*} Y_{i^*,2})^{1/(x+\eta_{i^*}-w^*)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w^*)})} \right)^{s_{w^*,1}^* \phi'^* + s_{w^*,2}^*}$$
$$\cdot e(C_1, ((Y_{j,1} Y_{i,1}^{-1})^{\phi'} (Y_{j,2} Y_{i,2}^{-1}))^{1/(x+\eta_i - w)}) \left(\frac{C_2}{e((C_1, g)^{1/(x+\eta_i-w)})} \right)^{s_{w,1}\phi' + s_{w,2}}.$$

If $C_1 \neq C_1^*$ or $C_2 \neq C_2^*$ or $w \neq w^*$ or $\eta_i \neq \eta_{i^*}$, since $s_{w^*,k}^*$ is uniformly random and independent from $\mathcal{A}$'s view, then the re-encryption result C_3' is a random value from $\mathcal{A}$'s view. Thus it will not leak the bit β.

If $C_1 = C_1^*, C_2 = C_2^*, w = w^*, \eta_i = \eta_{i^*}$ and $C_3 = C_3^*$, this situation can be discussed in two cases:

- If $C_0 = svk \equiv svk^* = C_0^*$, then

  $$(C_1, C_2, C_3, C_4, C_5, \sigma) \neq (C_1^*, C_2^*, C_3^*, C_4^*, C_5^*, \sigma^*)$$

  $\mathcal{B}$ is faced with an occurrence of F_{OTS} and halts.

- If $C_0 = svk \neq svk^* = C_0^*$, we have $\phi \neq \phi^*$ and $\phi' \neq \phi'^*$,

$$C_3' = R^* e(C_1^*, (Y_{i^*,1}^{\phi'^*} Y_{i^*,2})^{1/(x+\eta_{i^*}-w^*)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w^*)})}\right)^{s^*_{w^*,1}\phi'^* + s^*_{w^*,2}}$$
$$\cdot e(C_1, ((Y_{j,1}Y_{i,1}^{-1})^{\phi'}(Y_{j,2}Y_{i,2}^{-1}))^{1/(x+\eta_i-w)}) \left(\frac{C_2^*}{e((C_1^*, g)^{1/(x+\eta_{i^*}-w^*)})}\right)^{s^*_{w^*,1}\phi' + s^*_{w^*,2}}.$$

  Since $\phi' \neq \phi'^*$, then $l = s^*_{w^*,1}\phi' + s^*_{w^*,2}$ is uniformly random and independent from $l^* = s^*_{w^*,1}\phi'^* + s^*_{w^*,2}$. Thus it will not leak the bit β.

This completes the proof of Lemma 2. □

Lemma 3. *If there exists an IND-L1-CCA adversary $\mathcal{A}$ against our scheme, then there exists an algorithm $\mathcal{B}$, which can solve the DBDH problem.*

Proof. In our scheme, the challenge ciphertext of non-transformable ciphertext security is the same as the transformed ciphertext security. Therefore, without losing generality, we only consider the proof of the non-transformable ciphertext security. We first let the challenger set the groups $\mathbb{G}_1$ and $\mathbb{G}_2$ with an efficient bilinear map e and a generator g of $\mathbb{G}_1$. Simulator $\mathcal{B}$ inputs a DBDH instance (g, g^a, g^b, g^c, T), and has to distinguish $T = e(g,g)^{abc}$ from a random element in $\mathbb{G}_2$.

The random oracles H_0, H_1, H_2, and H_3 controlled by $\mathcal{B}$ are the same as those in Lemma 1.

(1) Setup: Let λ be the security parameter and $(p, g, \mathbb{G}_1, \mathbb{G}_2, e)$ be the bilinear map parameters. Let message space be $\mathcal{M} = \{0,1\}^k$ and condition space be $\mathcal{W} = \mathbb{Z}_p^*$. The global system parameters are $(p, g, \mathbb{G}_1, \mathbb{G}_2, e, k, H_0, H_1, H_2, H_3, sig)$.

(2) Query phase 1. $\mathcal{A}$ makes the following queries:
- Uncorrupted key generation query $\langle i \rangle$: public keys of honest user $i \in HU$ are defined as the following: $\mathcal{B}$ selects a random value $x_i, \eta_{i,1}, \eta_{i,2}, y_{i,3}, y_{i,4} \in \mathbb{Z}_p^*$, computes $X_i = g^{x_i}$, $Y_{i,k} = g^{a+\eta_{i,k}}$. This implicitly defines the secret key value as $x_i = x_i$, $\{y_{i,k} = a + \eta_{i,k}\}_{k\in\{1,2,3,4\}}$. $\mathcal{B}$ sets target user's public key as $pk_{i^*} = (X_i, \{Y_{i,k}\}_{k\in\{1,2,3,4\}})$, and sends public key to $\mathcal{A}$.
- Corrupted key generation query $\langle i \rangle$: Public keys of corrupt user $i \in CU$ are the same as the key generation algorithm, this means the simulator $\mathcal{B}$ can know the both the public key and secret key of user $i \in CU$, and then sends (pk_i, sk_i) to $\mathcal{A}$.
- Re-encryption key query $\langle pk_i, w, pk_j \rangle$: since $\mathcal{B}$ can know the secret key part x_i for user i, $\mathcal{B}$ can compute it correctly. Sends the re-encryption key $rk_{i,w,j} = \{d_{w,k}, s_{w,k}\}_{k\in\{1,2\}}$ to $\mathcal{A}$.
- Trapdoor query $\langle pk_i, w \rangle$: since $\mathcal{B}$ can know the secret key part x_i for user i, $\mathcal{B}$ can compute it correctly. Sends the trapdoor $T_{i,w} = \{d_{w,k}, s_{w,k}\}_{k\in\{3,4\}}$ to $\mathcal{A}$.
- Decryption query$\langle pk_j, CT_j \rangle$: If $\langle pk_j, CT_j \rangle$ denote the queries on re-encrypted ciphertext (second level ciphertext), $CT_j = (C_0, C_2, C_3, C_4)$. For a user's $j \in CU$, $\mathcal{B}$ can decrypt it correct, since $\mathcal{B}$ knows the secret key for user $j \in CU$. For a user's $j \in HU$, For a user's $j \in HU$, then $\mathcal{B}$ searches for H_0^{List}, H_1^{List} and H_2^{List} to see whether there exists a tuple (C_0, C_2, C_4, ϕ'), a tuple (m, R, r) and a tuple (R, ω) such that

$$C_2 = e(g,g)^r, \qquad C_3 = e(g, Y_{j,1})^{r\phi'} e(g, Y_{j,2})^r R, \qquad C_4 = m \oplus H_2(R).$$

  If yes, it outputs m to $\mathcal{A}$; else it outputs $\perp$.

(3) Challenge. Once $\mathcal{A}$ decides that Phase 1 is over, it outputs a target public key pk_{i^*}, and two equal length plaintexts (m_0, m_1). $\mathcal{B}$ responds by choosing a random $\beta \in \{0,1\}$, picks $R^* \in \mathbb{G}_2^*$, $C_0^* \in \{0,1\}^{k_1}$ and computes

$$C_2^* = e(g^b, g^c), \qquad C_4^* = m_\beta \oplus H_2(R^*).$$
$$\phi'^* = H_0(C_0^*, C_2^*, C_4^*), \qquad C_3^* = (T)^{(\phi'^*+1)} \cdot e(g^b, g^c)^{(\eta_{i^*,1}\phi'^* + \eta_{i^*,2})} R^*.$$

  To see this, let $H_1(m_\beta, R^*) = r^* = bc$, if $T = e(g,g)^{abc}$, then

$$C_2^* = e(g^b, g^c) = e(g,g)^{r^*}$$
$$C_3^* = (e(g,g)^{abc})^{(\phi'^*+1)} \cdot e(g^b, g^c)^{(\eta_{i^*,1}\phi'^* + \eta_{i^*,2})} R^*$$
$$= e(g,g)^{(a+\eta_{i^*,1})bc\phi'^*} e(g,g)^{(a+\eta_{i^*,2})bc} R^* = e(g, Y_{i^*,1})^{r^*\phi'^*} e(g, Y_{i^*,2})^{r^*} R^*.$$

(4) Query phase 2. $\mathcal{A}$ continues making queries as in the query phase 1.

(5) Guess. $\mathcal{A}$ outputs the guess β', if $\beta' = \beta$, then output 1 meaning $T = e(g,g)^{abc}$; else output 0 meaning $T = e(g,g)^r$.

Probability analysis: Suppose there exists a polynomial-time adversary, $\mathcal{A}$, in Game 2 that can attack our scheme with an advantage ε. Now we provide the probability of the simulator $\mathcal{B}$:

When $T = e(g,g)^{abc}$ then $\mathcal{A}$ must satisfy $|Pr[\beta' = \beta] - 1/2| \geq \varepsilon$. When T is uniform in $\mathbb{G}_2$, R^* and C_3^* are uniformly random and independent, then $Pr[\beta' = \beta] = 1/2$. Therefore, when a, b, c are uniform in $\mathbb{Z}_p^*$ and T is uniform in $\mathbb{G}_2$, we have that $|Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] - Pr[\mathcal{B}(g, g^a, g^b, g^c, e(g,g)^r) = 1]| \geq |(1/2 \pm \varepsilon) - 1/2| = \varepsilon$ as required. This completes the proof of Lemma 3. □

Table 1
Comparison among various C-PRE and PRES schemes.

Scheme		WYTDB [34]	SCLL [31]	Ours
Cost	Enc1	$3t_e$	$4t_e + 2t_p + t_s$	$2.5t_e$
	Enc2	$3t_e + 1t_p$	$4t_e + 2t_p + t_s$	$5.5t_e + t_s$
	ReEnc	$3t_p$	$t_e + 4t_p + t_v$	$2t_e + t_p + t_v$
	Dec1	$2t_e + 1t_p$	$t_e + 5t_p + t_v$	$2t_e$
	Dec2	$2t_e + 3t_p$	$t_e + 5t_p + t_v$	$3t_e + t_v$
Length	pk	$1\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$	$1\|\mathbb{G}_1\|$	$5\|\mathbb{G}_1\| + 1\|\mathbb{G}_2\|$
	sk	$1\|\mathbb{Z}_p^*\|$	$1\|\mathbb{Z}_p^*\|$	$5\|\mathbb{Z}_p^*\|$
	$Level_1$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_2\| + \|\mathcal{M}\|$	$\|svk\| + 3\|\mathbb{G}_1\| + 2\|\mathbb{G}_2\| + \|\sigma\|$	$\|svk\| + 2\|\mathbb{G}_2\| + \|\mathcal{M}\|$
	$Level_2$	$2\|\mathbb{G}_1\| + \|\mathbb{G}_2\| + \|\mathcal{M}\| + \|\mathcal{W}\|$	$\|svk\| + 3\|\mathbb{G}_1\| + 2\|\mathbb{G}_2\| + \|\sigma\|$	$\|svk\| + \|\mathbb{G}_1\| + 3\|\mathbb{G}_2\| + \|\mathcal{M}\| + \|\sigma\|$
	ReKey	$2\|\mathbb{G}_1\|$	$\|\mathbb{Z}_p^*\|$	$2\|\mathbb{G}_1\| + 2\|\mathbb{Z}_p^*\|$
	Trapdoor	–	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\| + 2\|\mathbb{Z}_p^*\|$
Security		CCA	CCA	CCA
ROM		Yes	Yes	Yes
Keyword search		No	Yes	Yes
Anonymity		No	Yes	Yes
Unidirectionality		Yes	No	Yes
Collusion-resistant		Yes	No	Yes

4. Performance comparison

In this section, we compare our schemes with Shao et al.'s proposed bidirectional proxy re-encryption with keyword search scheme (we denote it as SCLL) [31] and Weng et al.'s proposed CCA secure C-PRE (we denote it as WYTDB) [34]. Since Weng et al.'s first C-PRE (WDCL) [33] is not CCA secure and Weng et al.'s proposed CCA secure C-PRE is the most efficient scheme in the known C-PRE, we only include WYTDB for our comparison. Let $|\mathcal{M}|$, $|\mathbb{G}_1|$, $|\mathbb{G}_2|$, $|svk|$ and $|\sigma|$ denote the bit-length of a plaintext, an element in groups $\mathbb{G}_1$ and $\mathbb{G}_2$, the verification key and signature of one-time signature, respectively. We denote t_p, t_e, t_s, and t_v as the computational cost of a bilinear pairings, an exponentiation over a bilinear group, a one-time signature and verification, respectively. Notice that encryption in our scheme does not require any pairing computations once $e(g, g)$ and $e(g, Y_{i,k})$ have been viewed as public key. Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be the bilinear groups and svk and σ be the one-time signatures public key and signature. The result of the comparison is outlined in Table 1.

From Table 1, it is observed that our C-PRES from Section 3 outperforms Shao et al.'s PRES scheme (SCLL) in terms of both computational and communication costs. Furthermore, ours is collusion resistant as well as it is a conditional re-encryption scheme, while Shao et al.'s PRES scheme is bidirectional only. Our C-PRE is also superior to Weng et al.'s scheme (WYTDB) [34] in all terms of computation costs. More importantly, our scheme provides anonymity, in contrast to Weng et al.'s scheme. However, like Weng et al.'s scheme [34], our scheme is limited in that its security relies on the random oracle.

5. Applications of C-PRES

In this section, we provide three applications for C-PRES schemes. We specifically select these applications to demonstrate how C-PRES schemes can be used to present solutions in these scenarios.

5.1. Application in privacy-preservation of online photo sharing

Online photo sharing is an increasingly popular function of social-networking services, allowing users to share their photos with family and friends privately. By definition, photo sharing is the process of publishing or transferring a user's digital photos online, thus enabling the user to share them with others privately. This function is provided through both websites and applications that facilitate the upload and display of images. The term can also be loosely applied to the use of online photo galleries that are set up and managed by individual users. Sharing means that other users can view some of the pictures according to the access right. It allows users to set up privacy policies to control who can access their photos. With the increased popularity of mobile devices, such as iPhone and Blackberry, users can comfortably update their photos wherever they go and the service enables them to notify their peers automatically, once the peers are authorized by sufficient access policies. The service enables photo-sharing providers to disseminate users' photo data in a secure manner, since the data are actually encrypted. Further, the users can control when or where the data can be viewed by their peers. And the photo sharing provider can learn nothing about the data of the user including the tag, keyword and content of the photo.

A naive solution could be provided as follows. A user, Alice, could encrypt her photo prior to sending it to the photo sharing provider, and hence, protecting it from the provider or other adversaries. To enable her peers, Bob and Carol, to view her photo, Alice can securely disseminate her key to both Bob and Carol. Rather than using a common shared key, Alice could establish pair-wise secret keys with each of her friends or incorporate asymmetric keys, which both require a

great deal of additional storage, computation and communication overheads. Hence, although this solution is feasible, this is impractical.

Furthermore, since the data are actually encrypted, it needs a searchable encryption to ensure that the photo sharing provider can find the encrypted data (tag or keyword of the photo). We can solve this problem using C-PRES to provide the solution to the above problem, Alice sends a re-encryption key to Bob to share her location with Bob. The re-encryption key is computed using Bob's public key and Alice's private key, and then it is sent to the photo sharing service provider. When Bob would like to acquire Alice's location, he will first send a request to the location sharing provider. Then, the photo sharing provider will retrieve Alice's last encrypted photo and apply the re-encryption key and policies defined by Alice. Finally, the photo sharing provider will provide this information to Bob. Upon receiving this information, Bob can then decrypt the photo.

Alice may only want her families (Charlie) to see her photo when traveling, but she does not want others (David) to do so. To enable this kind of services, we will do the following.

First, let keyword $w = \text{“}Travel''$, and sends the trapdoor $T_{Alice,w}$ and conditional re-encryption key $rk_{Alice,w,Charlie}$ to the photo sharing provider. Then, the ciphertext containing Alice's photo, which is encrypted with the keyword $w = \text{“}Travel''$ can be re-encrypted by the provider to her families (Charlie) but not to her friends (David). Clearly, the provider can learn nothing about the user's data including the tag, keyword and content of the photo. C-PRES, though very efficient, is still too time-consuming to encrypt large volumes of data. To overcome this, the actual data is encrypted by a more efficient hybrid encryption scheme, where a secure symmetric encryption (E, D) is selected to be used to encrypt the photo data under a random key K, and subsequently, the random key K is then encrypted using the C-PRES scheme by $Enc2(pk_i, K, w)$. Then, the ciphertext would be in the form of $CT = c_1||c_2 = Enc2(pk_i, K, w)||E_K(m)$. As discussed in the introduction, if both encryptions are independent, then the hybrid scheme may not be CCA secure. Clearly, if there is no link between the symmetric encryption and the second encryption, then the adversary can obtain a new ciphertext by $CT' = c_1||c_3$ where c_3 is randomly chosen. The adversary can ask for a re-encryption query of $CT' = c_1||c_3$ for a corrupted delegate, the proxy will re-encrypt it since the proxy cannot check the validity of the new ciphertext $CT' = c_1||c_3$. To overcome this issue, we could include the symmetric encryption of data on input to the hash function as $\phi' = H_0(C_0, C_2, C_4, E_K(m))$ and the one time signature of the symmetric encryption of data.

5.2. Application in personal health record

Consider the scenario in a Personal Health Record (PHR) disclosure. A PHR contains all kinds of health-related information about an individual (say, Alice). For example, a PHR contains medical history that includes surgery, illness, laboratory test results, allergies, chronic diseases, vaccinations, imaging (X-ray) reports, immunization records, etc. and the sensitive information provided by Alice including her age, weight, family, food statistics, contact information and any other information related to her health. It is clear that a PHR contains very sensitive data that must be protected. The remote PHR data centers are responsible for storing users' PHR data including the data need to be protected. On the one hand, the remote data centers are usually assumed to be semi-trusted, so the private data should be encrypted. On the other hand, users usually do not retrieve all the encrypted data but part of them, which demands the searchable encryption scheme supporting the keyword-based search on the ciphertext.

To ensure Alice's privacy, one may decide to encrypt her PHR and store only the ciphertexts in the PHR database. Then, the database can be decrypted on demand. This solution is not practical since Alice needs to be involved in every request to conduct the decryption.

Incorporating a proxy re-encryption would be a viable solution in this situation. Nevertheless, the traditional proxy re-encryption is not suitable since the proxy who has the re-encryption key can convert all ciphertext of PHR. Thus, Ibraimi, Tang, Hartel, and Jonker use Type-based Proxy Re-Encryption (same as C-PRE) to get a more fine-grained PHR disclosure scheme. But the limitation of C-PRE is the C-PRE will leak the information of keyword (or type) to remote PHR data centers.

It will be great if we choose C-PRES, as the situation will change completely. Suppose different categories of Alice's encrypted PHR are accompanied with a keyword, such as the encrypted PHR under the keyword “*allergies*”, or the encrypted PHR under the keyword “*imaging reports*”. Then, Alice categorizes her PHR according to her privacy concerns. For instance, she can send a trapdoor $T_{Alice,w}$ and a re-encryption key $rk_{Alice,w,Practitioner}$ from Alice to a general practitioner under the keyword $w = \text{“}allergies''$ to the proxy P. By doing this, for the encrypted PHR under the keyword “allergies”, it can be searchable and re-encrypted by proxy P, then it can be decrypted by the specifically authorized general practitioner.

5.3. Application in wireless sensor networks

Recent advancements in wireless communications and electronics have enabled rapid development of wireless sensor network (WSN). WSN has become the most attractive technology for building automation, which allows users to designate the building network and subsequently, control building appliances depending on their needs. We assume that the data of building automation are stored in remote data centers and that the user can access the data through the Internet.

Nevertheless, messages exchanged through the WSN and remote data centers must be protected to improve the safety of building automation. Further, its importance is to provide various services to authorized users or would-be users, such as visitors.

The remote data centers are responsible for storing users' data including the data need to be protected. On the one hand, the remote data centers are usually assumed to be semi-trusted, so the private data should be encrypted. On the other hand, users usually do not retrieve all the encrypted data but only a part of them, which requires the searchable encryption scheme to support the keyword-based search and keyword-based re-encryption on the ciphertext.

We intend to apply an anonymous conditional proxy re-encryption with keyword search, which re-encrypts a ciphertext of the visitors to delegate the capability of decryption to offer a practical and viable solution for WSN.

At first, visitors cannot control anything in the house due to the security policy. For this reason, the owner needs to determine the delegation power suitable for visitors to utilize the appliances. After authorization, appliances (lights, electronic faucets) can now be controlled by visitors. All of the appliances in the building can be managed by the owner and visitors in C-PRES scheme. For example, the owner may want the visitor to control lights, but not the electronic window curtains. We use C-PRES to solve this problem as follows. First, let the keyword $w =$ "*lights*" and send the trapdoor $T_{Owner,w}$ and re-encryption key $rk_{Owner,w,Visitor}$ to the remote data centers. Then, the ciphertext of lights data which is encrypted by keyword $W =$ "*lights*" can be searched and re-encrypted by remote data centers who has $rk_{Owner,W,Visitor}$ and send to the visitor, and therefore, they can be decrypted by the visitor. Nevertheless, the visitor cannot re-encrypt the ciphertext of electronic window curtains data which are encrypted by keyword $W =$ "*curtains*". Clearly, the provider can learn nothing about the user's data such as the specific type and location of the appliance that is being operated.

6. Conclusion

In this paper, we present *the first* Chosen-Ciphertext Secure anonymous conditional proxy re-encryption with keyword search (C-PRES) scheme, which is an affirmative and effective answer to the open question posed in [33] and [31]. Our scheme offers several advantages over previous such systems, including: chosen-ciphertext security; keyword-anonymity; unidirectionality; non-interactivity; and collusion-resistance.

This work motivates a few interesting questions. The first one is how to construct a CCA-secure C-PRES scheme without random oracles. Actually, our scheme is rely on the ROM since we use Fujisaki–Okamoto transformation to ensure the CCA secure of first level security, it seems possible to adopt the secure pseudo-random number generators instead of Fujisaki–Okamoto transformation to avoid the ROM, we leave it as further work. Then, the second question is how to construct C-PRES schemes without pairings.

Acknowledgment

This work is supported by ARC Future Fellowship FT0991397.

References

[1] G. Ateniese, K. Fu, M. Green, S. Hohenberger, Improved proxy re-encryption schemes with applications to secure distributed storage, in: Proc. of the 12th Annual Network and Distributed System Security Symposium, 2005, pp. 29–44.
[2] M. Abramowitz, I.A. Stegun. (Eds.), Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables, 9th printing, Dover, New York, 1972.
[3] J. Baek, R. Safavi-Naini, W. Susilo, Public key encryption with keyword search revisited, in: Proc. of Applied Cryptography and Information Security 06, ACIS 2006, in: LNCS, vol. 5072, Springer, Heidelberg, 2008, pp. 1249–1259.
[4] J. Baek, R. Safavi-Naini, W. Susilo, On the integration of public key data encryption and public key encryption with keyword search, in: Proc. of ISC 2006, in: LNCS, vol. 4176, Springer, Heidelberg, 2006, pp. 217–232.
[5] M. Blaze, G. Bleumer, M. Strauss, Divertible protocols and atomic proxy cryptography, in: Proc. of EUROCRYPT 1998, in: LNCS, vol. 1403, Springer, Heidelberg, 1998, pp. 127–144.
[6] D. Boneh, X. Boyen, Efficient selective-ID based encryption without random oracles, in: Proc. of EUROCRYPT 2004, in: LNCS, vol. 3027, Springer, Heidelberg, 2004, pp. 223–238.
[7] D. Boneh, G. Di Crescenzo, R. Ostrovsky, G Persiano, Public key encryption with keyword search, in: Proc. of EUROCRYPT 2004, in: LNCS, vol. 3027, Springer, Heidelberg, 2004, pp. 506–522.
[8] J.W. Byun, H.S. Rhee, H.A. Park, D.H. Lee, Off-line keyword guessing attacks on recent keyword search schemes over encrypted data, in: Proc. of SDM 2006, in: LNCS, vol. 4165, Springer-Verlag, 2006, pp. 75–83.
[9] R. Canetti, S. Hohenberger, Chosen-ciphertext secure proxy re-encryption, in: Proc. of the 14th ACM Conference on Computer and Communications Security, ACM, New York, NY, USA, 2007, pp. 185–194.
[10] R. Canetti, S. Halevi, J. Katz, Chosen-ciphertext security from identity-based encryption, in: EUROCRYPT 2004, in: LNCS, vol. 3027, Springer, Heidelberg, 2004, pp. 202–222.
[11] R. Canetti, H. Krawczyk, J.B. Nielsen, Relaxing chosen-ciphertext security, in: CRYPTO 2003, in: LNCS, vol. 2729, Springer, Heidelberg, 2003, pp. 565–582. Full version: Cryptology ePrint Archive, Report 2003/174 (2003). http://eprint.iacr.org/.
[12] C. Chu, W. Tzeng, Identity-based proxy re-encryption without random oracles, in: Proc. of ISC 2007, in: LNCS, vol. 4779, Springer, Heidelberg, 2007, pp. 189–202.
[13] C. Chu, J. Weng, S. Chow, J. Zhou, R. Deng, Conditional proxy broadcast re-encryption, in: Proc. of ACISP 2009, in: LNCS, vol. 5594, Springer, Heidelberg, 2009, pp. 327–342.
[14] S. Chow, J. Weng, Y. Yang, R. Deng, Efficient unidirectional proxy re-encryption, in: Proc. of AFRICACRYPT 2010, in: LNCS, vol. 6055, Springer, Heidelberg, 2010, pp. 316–332.
[15] R. Deng, J. Weng, S. Liu, K. Chen, Chosen-cipertext secure proxy re-encryption without pairings, in: Proc. of CANS 2008, in: LNCS, vol. 5339, Springer, Heidelberg, 2008, pp. 1–17.
[16] L. Fang, W. Susilo, J. Wang, Anonymous conditional proxy re-encryption without random oracle, in: Proc. of ProvSec 2009, in: LNCS, vol. 5848, Springer, Heidelberg, 2009, pp. 47–60.

[17] E. Fujisaki, T. Okamoto, Secure integration of asymmetric and symmetric encryption schemes, in: Proc. of CRYPTO 1999, in: LNCS, vol. 1666, Springer, Heidelberg, 1999, pp. 537–554.
[18] C. Gentry, Practical identity-based encryption without random oracles, in: Proc. of EUROCRYPT 2006, in: LNCS, vol. 4004, Springer, Heidelberg, 2006, pp. 457–464.
[19] M. Green, G. Ateniese, Identity-based proxy re-encryption, in: Proc. of ACNS 2007, in: LNCS, vol. 4521, Springer, Heidelberg, 2007, pp. 288–306. Full version: Cryptology ePrint Archieve: Report 2006/473.
[20] P. Golle, J. Staddon, B. Waters, Secure conjunctive search over encrypted data, in: M. Jakobsson, M. Yung, J. Zhou (Eds.), Proc. of Second International Conference on Applied Cryptography and Network Security, ACNS 2004, in: LNCS, vol. 3089, Springer-Verlag, 2004, pp. 31–45.
[21] S. Hohenberger, G.N. Rothblum, A. Shelat, V. Vaikuntanathan, Securely obfuscating re-encryption, Journal of Cryptology. Express 24 (2) (2011) 694–719.
[22] I.R. Jeong, J.O. Kwon, D. Hong, D.H. Lee, Constructing PEKS schemes secure against keyword guessing attacks is possible? Computer Communications. Express 32 (2) (2009) 394–396.
[23] B. Libert, D. Vergnaud, Unidirectional chosen-ciphertext secure proxy re-encryption, in: Proc. of PKC 2008, in: LNCS, vol. 4939, Springer, Heidelberg, 2008, pp. 360–379.
[24] B. Libert, D. Vergnaud, Unidirectional chosen-ciphertext secure proxy re-encryption, IEEE Transactions on Information Theory 57 (3) (2011) 1786–1802. Full version of : http://hal.inria.fr/inria-00339530/en/.
[25] D.J. Park, K. Kim, P.J. Lee, Public key encryption with conjunctive field keyword search, in: C.H. Lim, M. Yung (Eds.), Proc. of Information Security Applications, 5th International Workshop, WISA 2004, in: LNCS, vol. 3325, Springer-Verlag, 2005, pp. 73–86.
[26] T. Matsuda, R. Nishimaki, K. Tanaka, CCA proxy re-encryption without bilinear maps in the standard model, in: Proc. of PKC 2010, in: LNCS, vol. 6056, Springer, Heidelberg, 2010, pp. 261–278.
[27] H.S. Rhee, J.H. Park, W. Susilo, D.H. Lee, Improved searchable public key encryption with designated tester, in: Proc. of the 4th international Symposium on information, Computer, and Communications Security, ASIACCS 2009, ACM, New York, NY, 2009, pp. 376–379.
[28] H.S. Rhee, W. Susilo, H-J. Kim, Secure searchable public key encryption scheme against keyword guessing attacks, IEICE Electron. Express 6 (5) (2009) 237–243.
[29] H.S. Rhee, J.H. Park, W. Susilo, D.H. Lee, Trapdoor security in a searchable public-key encryption scheme with a designated tester, Journal of Systems and Software 83 (5) (2010) 763–771.
[30] J. Shao, Z. Cao, CCA-secure proxy re-encryption without pairings, in: Proc. of PKC 2009, in: LNCS, vol. 5443, Springer, Heidelberg, 2009, pp. 357–376.
[31] J. Shao, Z. Cao, X. Liang, H. Lin, Proxy re-encryption with keyword search, Information Sciences. Express 180 (13) (2010) 2576–2587.
[32] Q. Tang, Type-based proxy re-encryption and its construction, in: Proc. of INDOCRYPT 2008, in: LNCS, vol. 5365, Springer, Heidelberg, 2008, pp. 130–144.
[33] J. Weng, R. Deng, C. Chu, X. Ding, J. Lai, Conditional proxy re-encryption secure against chosen-ciphertext attack, in: Proc. of the 4th International Symposium on ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, 2009, pp. 322–332.
[34] J. Weng, Y. Yang, Q. Tang, R. Deng, F. Bao, Efficient conditional proxy re-encryption with chosen-ciphertext security, in: Proc. of the 12th International Conference on Information Security, ISC 2009, 2009, pp. 151–166.
[35] B. Waters, D. Balfanz, G. Durfee, D. Smetters, Building an encrypted and searchable audit log, in: Proc. of Network and Distributed System Security Symposium, NDSS 2004, 2004.
[36] W.C. Yau, S.H. Heng, B. Goi, Off-line keyword guessing attacks on recent public key encryption with keyword search schemes, in: Proc. of ATC 2008, in: LNCS, vol. 5060, Springer-Verlag, 2008, pp. 100–105.
[37] W.C. Yau, R. Phan, S.H. Heng, B. Goi, Proxy re-encryption with keyword search: new definitions and algorithms, Journal of Security and Its Applications. Express 5 (2) (2011) 149–160.
[38] B. Zhang, F. Zhang, An efficient public key encryption with conjunctive-subset keywords search, Journal of Network and Computer Applications. Express 34 (1) (2011) 262–267.
[39] R. Zhang, H. Imai, Generic combination of public key encryption with keyword search and public key encryption, in: Proc. of Cryptology and Network Security, 6th International Conference, CANS 2007, in: LNCS, vol. 4856, Springer, Heidelberg, 2007, pp. 159–174.
[40] R. Zhang, G. Hanaoka, J. Shikata, H. Imai, On the security of multiple encryption or CCA-security+CCA-security=CCA-security? in: Proc. of PKC 2004, in: LNCS, vol. 2947, Springer, Heidelberg, 2004, pp. 360–374.